UNIVERSIDAD DE LA RIOJA

TRABAJO FIN DE ESTUDIOS

Título

Personalización de vídeo en tiempo real

Autor/es

AIDA IRUZUBIETA MARTÍNEZ

Director/es

ELOY JAVIER MATA SOTÉS

Facultad

Facultad de Ciencia y Tecnología

Titulación

Grado en Ingeniería Informática

Departamento

MATEMÁTICAS Y COMPUTACIÓN

Curso académico

2016-17

publicaciones.unirioja.es
E-mail: publicaciones@unirioja.es

UNIVERSIDAD DE LA RIOJA

Facultad de Ciencia y Tecnología

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática

Personalización de vídeo en tiempo real

Alumno:

Aida Iruzubieta Martínez

Tutores:

Eloy Javier Mata

Logroño, junio, 2017

Resumen

La personalización de vídeo en tiempo real consiste en la reproducción de un vídeo a mismo tiempo que se edita en un segundo plano. Tras editar el vídeo, sin pausar la reproducción del mismo, se comienza a ver el vídeo personalizado.
Esto es posible porque el envío del vídeo se hace por fragmentos y solo se edita a partir de determinados fragmentos.

El desarrollo de este proyecto se ha llevado a cabo estudiando las diferentes tecnologías y protocolos que se han encontrado durante determinado periodo de investigación y posteriormente, la implementación de una prueba de concepto que demuestra lo anteriormente estudiado, se han utilizado las herramientas Ffmpeg y MP4Box para el principal desarrollo junto con el protocolo de streaming adaptativo DASH (Dynamic Adaptive Streaming over HTTP).

Abstract

Customizing video in real time consist of the playing of a video while editing in the background. After that, and without interrupting the video, the personalized one starts. Therefore, this is possible because the video is sent by pieces and it is edited only in certain fragments.

The development of this project has been carried out by studying the different technologies and protocols that have been found during a certain period of investigation and the following implementation of a proof that demonstrates what has been previously studied. For this, tools as Ffmpeg and MP4Box have been used for the main development along with the Adaptive Streaming Protocol DASH (Dynamic Adaptive Streaming over HTTP).

Índice

➢ Capítulo 1. Documentos Objetivos del Proyecto (DOP)

1. Introducción

a) Contexto

La empresa en la que se desarrollará el Trabajo de Fin de Grado es Viwom, empresa dedicada a crear campañas publicitarias de vídeo embebido en el email.

Cuando una empresa desea hacer campañas publicitarias sobre alguno de sus productos trabajando con Viwom, tiene dos opciones:

- Crear un vídeo publicitario y una plantilla HTML y subirlos al servidor de ficheros
- Encargar el vídeo a Viwom que entonces el mismo crea el vídeo publicitario y la plantilla HTML y los aloja en el servidor de ficheros.

Cuando se sube el vídeo se genera:

- Una imagen representativa del vídeo.
- Un gif con determinadas imágenes del vídeo

La empresa propietaria del vídeo, comprueba si el gif y la imagen son de su gusto, si no, puede subir los suyos propios.

Cuando se tiene el vídeo y la plantilla HTML disponibles en el servidor, el servidor genera una nueva plantilla HTML con el vídeo integrado en ella, plantilla que después la empresa publicitadora enviará a sus clientes (clientes consumidores del vídeo) mediante su ESP (Email Service Provider)

En este proyecto se desea estudiar la forma en la que añadir a esta plataforma una funcionalidad que genere un vídeo personalizado en tiempo real.

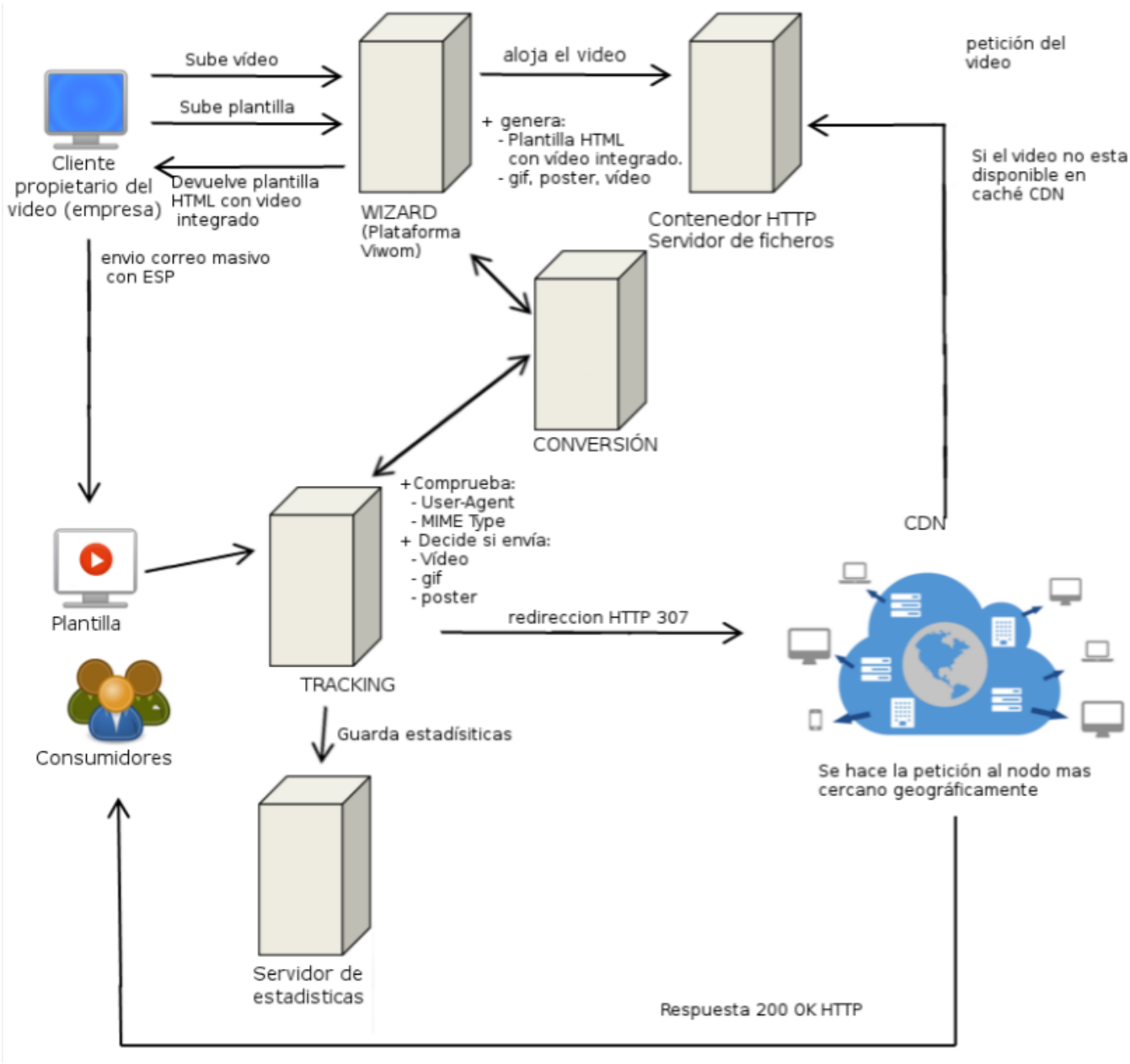


Figura1. Descripción del entorno

b) Antecedentes y justificación del proyecto

La forma actual en la que se envía un vídeo embebido en el email es de manera convencional, un vídeo para muchos usuarios, siendo el mismo para cualquier cliente consumidor, y a pesar de llegar hasta el cliente final de una manera muy directa, éste no ve que esa publicidad está hecha para él.

Ante la necesidad de crear un vídeo todavía más cercano al cliente, se desea realizar la personalización de vídeo en tiempo real.

En la primera etapa del marketing digital, también llamado web 1.0, las empresas eran las únicas que generaban contenido, de una forma única, dirigiéndolo a una gran audiencia sin selección, sin capacidad de respuesta. Se limitaba a mostrar un "catálogo" en línea (online).

En el marketing digital actual, también llamado web 2.0, los usuarios se convierten en consumidores y productores de información.

Cada vez se busca más llegar al usuario según sus gustos y características, investigando lo que le gusta y adaptando el contenido a él.

El objetivo es ayudar a los clientes a identificar mejor lo que quieren.

La personalización consiste en entregar un producto o servicio que satisfaga las necesidades del cliente, para ello se requiere el intercambio de información entre clientes y empresas, por lo que los clientes tienen que estar dispuestos a compartir información "personal" en cuanto a sus deseos y gustos.

Bajo la idea de hacer llegar la publicidad hasta el usuario, es decir, la publicidad busca el usuario y no el usuario busca la publicidad, nace Viwom.

Para hacer más fácil al cliente encontrar dicha publicidad y mucho más efectivo para las empresas que se publicitan mediante esta plataforma, que dicho vídeo sea visualizado por un mayor número de clientes.

2. Alcance

a) Objetivos

El principal objetivo del proyecto es estudiar las formas que hay de personalizar vídeo y escoger la forma que mejor se adapta a las necesidades de la empresa. Una vez encontrada, hacer una aplicación web (prueba de concepto), la cual, pasado un vídeo en el servidor web, cuando el consumidor del vídeo introduzca previamente unos parámetros a través de un formulario, automáticamente añada un parámetro de texto al vídeo.

Por simplicidad supondremos que ya tenemos el vídeo subido en el servidor. En un caso real, el vídeo lo subiría un cliente de Viwom (empresa que quiere publicitar sus productos).

Siguiendo determinados parámetros de personalización, como pueden ser: el texto que debe aparecer en el vídeo, (el momento del vídeo en el que aparece el texto, durante cuánto tiempo, en que posición, con que formato etc.)

Ese texto será información personal del usuario que lo reproduce (consumidor del vídeo), además, el vídeo debe empezar a reproducirse en el momento que el usuario le dé a play, teniendo el mínimo retardo posible, para que la personalización del mismo sea lo más transparente posible para él.

Aunque no es un objetivo inicial de este proyecto en caso de tener tiempo suficiente en su desarrollo, se hará como extra la interfaz del cliente que sube el vídeo al servidor web.

a) Personal/ Recursos Humanos

El personal implicado en este Proyecto de Fin de Grado será:

- Aida Iruzubieta: Ejecutora y responsable del proyecto.
- Eloy Mata: tutor de la universidad
  - Se dedicará a corregir aspectos referentes a la documentación y la forma en la que se desarrolla el proyecto a lo largo del tiempo estipulado para hacerlo y guiarme a cerca del desarrollo del mismo.
- Jorge Garcés: tutor de la empresa
  - Se dedicará a corregir los aspectos técnicos referentes al desarrollo de la herramienta a desarrollar y dictar los requisitos funcionales de la aplicación.

b) Requisitos

- Funcionales
  - El servidor web atenderá las peticiones vía web.
  - Se editará el vídeo modificando el fragmento de vídeo que corresponda a los parámetros anteriormente indicados.
  - El vídeo se enviará al cliente en forma de fragmentos, mientras se realiza la personalización en los otros fragmentos.
  - La interfaz del cliente que visualiza el vídeo tendrá un reproductor HTML5 que lo reproducirá de manera transparente para el usuario, mientras lo va recibiendo por fragmentos.

- No funcionales
  - Se realizará una documentación de uso (incluida en la memoria)
  - Se utilizará el entorno de desarrollo Eclipse con un Plugin para el lenguaje GO.
  - Se utilizará un servidor con Ubuntu 16.04 (Linux) para alojar la página web.
  - El servidor web se implementará mediante GO.

c) Entregables

- Producto
  Se entregará una aplicación web escrita en GO compuesta por:
  - Script de transformación del vídeo: script escrito en GO que edita divide y sustituye los fragmentos de vídeo, es decir, realiza la personalización del vídeo.
  - Script de transmisión del vídeo: script escrito en GO que envía cada uno de los fragmentos por red, de manera que, en el reproductor del cliente que visualice el vídeo, se vean como si un solo fragmento fuera.
  - Interfaz web del cliente que tras enviar formulario visualiza el vídeo.

- Documentación / Planificación
  Se entregará un único documento.
  - Memoria del TFG: consiste en la memoria del proyecto, la cual se ha ido desarrollando mediante la realización del mismo, que contiene la información encontrada en el periodo de investigación, y la forma en la que se lleva a cabo el proyecto.

d) Plan de comunicaciones

- Reuniones: Se harán reuniones cara a cara con cada uno de los implicados en el proyecto, en diferentes momentos, reflejadas en el anexo de reuniones.
- Email: Se utilizará el email para concretar determinadas pautas o quedar para las reuniones.
- Teléfono: algunas de las reuniones no será posible tenerlas cara a cara y se tendrán que hacer por teléfono.
- Aplicación de comunicación interna de la empresa Viwomail: Mattermost
- Aplicación de seguimiento y control de la empresa: Trello.

e) Metodología

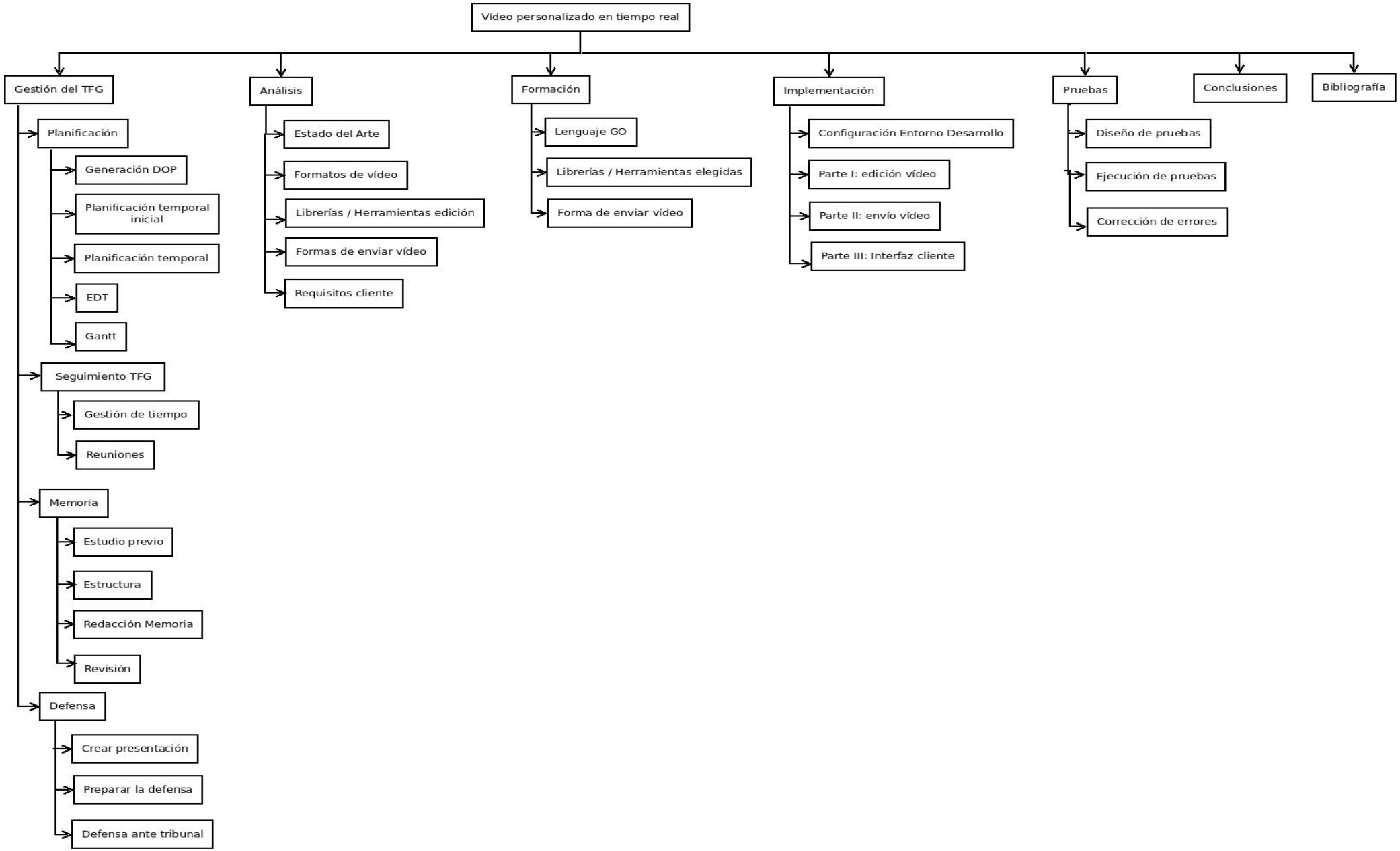
Para el desarrollo del proyecto, no se ha utilizado ninguna metodología concreta, ya que el proyecto tiene gran parte de investigación, y se irá haciendo a la vez que se aprenden nuevas tecnologías y funcionalidades.

Aun así, el trabajo se descompondrá en las típicas fases:

1. Análisis de requisitos
2. Estudio de herramientas y tecnologías.
3. Diseño
4. Implementación
5. Ejecución de pruebas
6. Verificación con el cliente
7. Mantenimiento

f) Planificación de tareas

EDT: Estructura de Desglose de Trabajo



- Tareas y descripción

Gestión del TFG
1. Planificación
1.1. Generación DOP (Documento Objetivos Proyecto)
En la generación del Documento Objetivos de Proyecto se definirán los diferentes aspectos relacionados con la planificación y objetivos del proyecto, es decir, lo que queremos conseguir una vez finalizado el tiempo de realización del proyecto.
1.2 Planificación temporal inicial
Se realizará una primera planificación estimada de las diferentes tareas del proyecto, al ser una gran parte del proyecto de investigación, se hará esta tarea, ya que no se sabe concretamente todas las tareas a realizar.
1.3 Planificación temporal
Una vez comenzado el proyecto, se permitirá modificar en cierto grado la planificación, quedando la definitiva para entregar con la memoria.
1.4 EDT
Se hará una estructura de desglose del trabajo, dividiendo de manera jerárquica las tareas a realizar.
1.5 Gantt
El diagrama de Gantt consiste en la planificación del tiempo que se estima que dure cada tarea, pero en forma de diagrama para que sea una manera más visual. Se hará con la aplicación de Libre Office Calc y una herramienta en versión de prueba.
2. Seguimiento TFG
2.1 Gestión de tiempo
Se hará una tabla escribiendo el tiempo estimado, el tiempo real invertido en cada tarea y la desviación del tiempo de cada una de ellas.
2.2 Reuniones
En esta tarea, se realizará unos puntos del día de la reunión a tener, a continuación, se hará un acta de la reunión con los temas tratados y los acuerdos que se han llegado.

Tabla1. Descripción de tareas

3. Memoria
3.1. Estudio previo
Se estudiarán las diferentes memorias de los alumnos que han desarrollado el TFG en años anteriores, que están disponibles en la página de la biblioteca de la Universidad de La Rioja, para ver la forma en la que se hace y se estructura la memoria para sacar ideas y conclusiones para hacer después la mía propia.
3.2. Estructura
Una vez miradas las memorias de otros alumnos, se hará una estructura para mi memoria, basado en el resto de memorias vistas, pero modificando determinadas cosas, para apropiarla a mi proyecto, dejando el índice hecho.
3.3. Redacción memoria
En esta tarea se pretende escribir la memoria al completo, aunque se irá escribiendo a lo largo de todo el proyecto, modificando algunos aspectos y añadiendo nuevos. Al ser un proyecto de investigación, se describirán ampliamente las tecnologías o protocolos utilizados o investigados para poder hacer las pruebas de concepto necesarias.
3.4. Revisión
Al finalizar el proyecto, se hará una revisión de la memoria al completo, se leerá y se modificarán las cosas que se crean que no están del todo bien, tras entregársela al tutor de la Universidad, es posible que se precisen nuevos cambios, por lo que se efectuarán en la revisión de la memoria.
4. Defensa
4.1. Crear presentación
Se creará una presentación, con el suficiente contenido para que me sirva de ayuda a la hora de presentar el proyecto realizado ante tribunal, adaptándola al tiempo de presentación estipulado.
4.2. Preparar defensa
Una vez creada la presentación, se invertirá un tiempo en preparar la presentación, que posteriormente se expondrá ante el tribunal, para prepararla, se debe tener claro cómo se va a organizar el tiempo de exposición, y hacer unos cuantos ensayos frente a alguien conocido, para cuando se llegue la presentación, tener claro y ensayado como se hará la presentación.
4.3. Defensa ante tribunal
Se expondrá el proyecto ante el tribunal, con la presentación hecha en las tareas anteriores y tal y como se ha ensayado antes.

Tabla1. (Continuación) Descripción de tareas

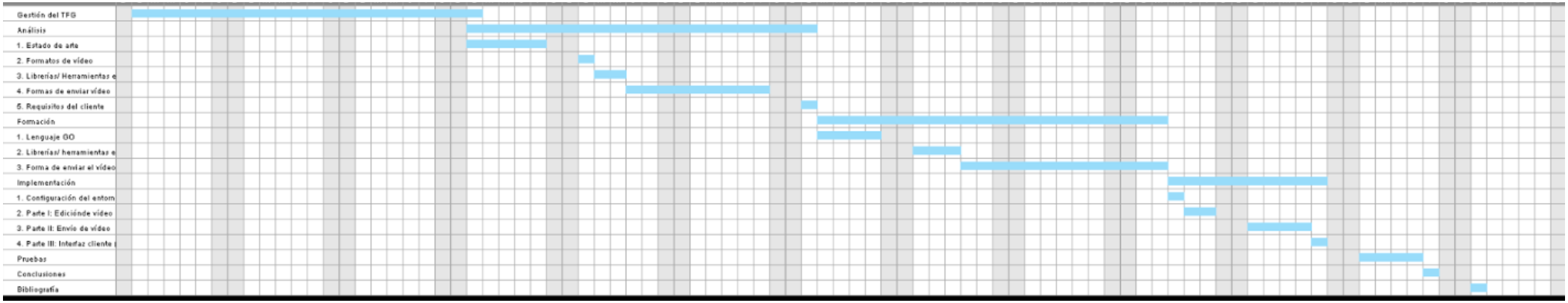
Análisis
1. Estado del Arte
Investigación de los casos de vídeo personalizado que hay hasta el momento, quién los hace y cómo. En cómo se hace se pretende entender la forma en la que lo personalizan, y que tecnologías se utilizan para ello. Se escribirán en la memoria los casos encontrados y si se elige alguno de ellos, o, por el contrario, se utiliza un método que se considere mejor que los existentes.
2. Formatos de vídeo
Estudio de los formatos de vídeo que se utilizan en la web, y cómo funcionan, se investigará también, como cambian los metadatos de los vídeos, dependiendo como se editen y las medidas que se deben tomar con ellos, para que la personalización de nuestro vídeo funcione correctamente con las tecnologías y métodos elegidos para llevar a cabo el proyecto.
3. Librerías / Herramientas de edición
Tras estudiar la forma en la que voy a personalizar el vídeo, se buscarán que librerías o herramientas para editar vídeo existen y se elegirán las más apropiadas para mi proyecto.
4. Formas de enviar vídeo
Una vez estudiado el estado de arte, se hará un estudio de las formas que existen actualmente de enviar vídeo por la red, eligiendo la más apropiada.
5. Requisitos del cliente
Redactar en la memoria los requisitos propuestos por el cliente, que en mi caso es el tutor de la empresa, y si algunos no estuvieran bien definidos por él, tomando algunas decisiones propias, escribir los propios requisitos.
Formación
1. Lenguaje GO
Estudio del lenguaje de programación GO, se hará un “curso” que hay en la página principal de GO para aprender algunos conceptos básicos, y después, cuando se empiece a realizar la aplicación, se profundizará en los aspectos más concretos para desarrollar la funcionalidad que deseo. En la formación de este lenguaje, recibiré ayuda de algunos de los compañeros de la empresa que trabajan con este lenguaje, y me pueden explicar ellos cosas más concretas que necesito desarrollar para mi proyecto.
2. Librerías / Herramientas elegidas
Una vez elegidas las herramientas para el desarrollo de la aplicación, se estudiará cómo funcionan y como se usan, concretamente para las funcionalidades que necesito
3. Forma de enviar el vídeo elegida
Una vez estudiadas las formas de enviar vídeo y elegida la mejor opción para el proyecto, se consultará con el cliente para que dé el visto bueno, se estudiará perfectamente y entenderá su funcionamiento, para después poder utilizarla. Se escribirá en la memoria los aspectos más importantes de esta forma elegida.

Tabla1. (Continuación) Descripción de tareas

Implementación
1. Configuración del entorno de desarrollo
Instalación y configuración del entorno de desarrollo elegido para desarrollar el proyecto, también se instalarán los diferentes plugin necesarios para poder desarrollar el proyecto con éxito y que me sea más fácil y cómodo desarrollarlo.
2. Parte I: edición de vídeo
Se desarrollará en el lenguaje indicado anteriormente, GO, un apartado que se encargue de editar el vídeo, de forma programada, pasando determinados parámetros de personalización. Pasado un vídeo y parámetros de personalización, devolverá otro vídeo personalizado, es decir, con los parámetros pasados, introducidos en el vídeo.
3. Parte II: envío de vídeo
Se desarrollará en el lenguaje GO, un apartado que se encargará de enviar el vídeo, de la forma anteriormente elegida.
4. Parte III: Interfaz cliente (reproductor)
Se desarrollará una pequeña página web, que contendrá un reproductor de vídeo en HTML5 capaz de reproducir el vídeo enviado de la manera anteriormente programada.
Pruebas
1. Diseño de pruebas
Se describen brevemente las pruebas que se van a llevar a cabo para comprobar que la aplicación funciona como se quiere.
2. Ejecución de pruebas
Se llevan a cabo las pruebas descritas en el apartado anterior, en caso de error, se pasa a la tarea siguiente.
3. Corrección de errores
Se dedicará un tiempo a corregir los posibles errores en la aplicación, una vez realizadas las pruebas necesarias, y en el momento de realizar las pruebas será cuando salgan los errores.
Conclusiones
Escribir una serie de conclusiones acerca de la realización del proyecto y de su contenido, que será una aclaración final de lo llevado a cabo a lo largo de todo el proyecto.
Bibliografía
Serie de enlaces, de los cuales se ha sacado la información para llevar a cabo el proyecto y escribir la memoria, situados al final de la memoria.

Tabla1. (Continuación) Descripción de tareas

- Gantt


Gestión del TFG
Análisis
1. Estado de arte
2. Formatos de vídeo
3. Librerías/ Herramientas e
4. Formas de enviar vídeo
5. Requisitos del cliente
Formación
1. Lenguaje GO
2. Librerías/ herramientas e
3. Forma de enviar el vídeo
Implementación
1. Configuración del entorn
2. Parte I: Edición de vídeo
3. Parte II: Envío de vídeo
4. Parte III: Interfaz cliente
Pruebas
Conclusiones
Bibliografía

- Periodo de ejecución

El periodo de ejecución del proyecto será entre la fecha de inicio 6 de febrero y fecha de depósito 21 de junio, teniendo días de no realización entre medio.
Los días marcados con el símbolo de prohibido, indican que no se ha trabajado en el proyecto.
Cada día se harán 5 horas, en horario de mañana, de 9:00h a 14:00h.

Un calendario de los días de trabajo es:

g) Planificación del tiempo

Tarea	Planificado
Gestión del TFG	80 horas
1.1. Generación DOP (Documento Objetivos Proyecto)	5 horas
1.2 Planificación temporal inicial	2,5 horas
1.3 Planificación temporal	3,5 horas
1.4 EDT	3 horas
1.5 Gantt	5 horas
2.1 Gestión de tiempo	3 horas
2.2 Reuniones	5 horas
3.1. Estudio previo	3 horas
3.2. Estructura	3 horas
3.3. Redacción memoria	30 horas
3.4. Revisión	6 horas
4.1. Crear presentación	5 horas
4.2. Preparar defensa	5 horas
4.3. Defensa ante tribunal	1 hora
Análisis	80 horas
1. Estado del Arte	25 horas
2. Formatos de vídeo	5 horas
3. Librerías / Herramientas de edición	10 horas
4. Formas de enviar vídeo	37 horas
5. Requisitos del cliente	3 horas
Formación	80 horas
1. Lenguaje GO	20 horas
2. Librerías / Herramientas elegidas	15 horas
3. Forma de enviar el vídeo elegida	45 horas
Implementación	40 horas
1. Configuración del entorno de desarrollo	5 horas
2. Parte I: edición de vídeo	10 horas
3. Parte II: envío de vídeo	20 horas
4. Parte III: Interfaz cliente (reproductor)	5 horas
Pruebas	20 horas
1. Diseño de pruebas	6 horas
2. Ejecución de pruebas	9 horas
3. Corrección de errores	5 horas
Conclusiones	2,5 horas
Bibliografía	5 horas

Tabla2. Planificación del tiempo.

➢ Capítulo 2. Análisis

1. Personalización de vídeo:

Se ha comenzado estudiando el estado del arte sobre la personalización de vídeo en tiempo real, encontrando diferentes sitios en los que se hace, e investigando la forma en la que llevan a cabo la personalización y se han encontrado diferentes opciones:

1. **Personalización de vídeo mediante superposición de JavaScript sobre el vídeo.** (un único vídeo almacenado en el servidor) JavaScript con diferentes parámetros para cada cliente. [1]

   Se tiene un vídeo común para todos los usuarios que hagan la petición, mediante un formulario, superpuesto a la página, mediante JavaScript, se piden determinados parámetros de personalización.

   Una vez se envía el formulario, el vídeo que está preparado para poner parámetros sobre él, los superpone en momentos concretos del vídeo mediante programación JavaScript.

   La personalización es inmediata, ya que no se necesita modificar el archivo del vídeo para ello, puesto que se hace en el momento de la reproducción mediante parámetros que se reproducen con JavaScript.

   Se sabe que es un único vídeo para todos con JavaScript, porque con la herramienta firebug se ha visto que el enlace al vídeo que nos muestra es un vídeo sin parámetros, con los huecos (dónde después se añade la personalización) en blanco.

   Conclusiones:

   ➢ Ventajas:
   - La personalización es inmediata, y fácil, sin modificación de vídeo, no es almacenado en disco.
   - Al hacerse la personalización en el lado del cliente, el servidor no es saturado por el proceso de cada petición, con lo que permite mayor número de peticiones simultáneas sin saturarse.

   ➢ Inconvenientes:
   - JavaScript actúa en el cliente y no en el servidor, por lo que necesitamos tener control sobre el cliente para poder llevar a cabo esta opción.
   - Al necesitar el cliente JavaScript, necesita que su navegador tenga la opción de ejecución de script activa, por lo que pierde seguridad en cuanto a ataques.

2. Personalización de vídeo completo.

a) Renderiza en el momento de la petición:

Con esperas en la web, almacena el vídeo (dividido en fragmentos) en el servidor durante un tiempo limitado, cada petición genera un vídeo. [2]

Previamente al vídeo se muestra un formulario al cliente para que este introduzca ciertos parámetros para posteriormente utilizarlos en la personalización, introduciéndolos en el vídeo. Estos parámetros de personalización pueden ser de texto y o para personalizar el vídeo, que dependiendo de estos aparezca un vídeo u otro, con partes comunes, aunque los parámetros sean diferentes.

Después de enviar el formulario aparece el vídeo, cuando el cliente pulsa play, tarda aproximadamente 1’ 15’’ en empezar a reproducirse (para un tamaño de vídeo de 2’).

Se supone que al pulsar play, el vídeo comienza a renderizarse, y hasta que éste no finaliza por completo, no se empieza a reproducir, lo que genera retardos y esperas al cliente consumidor del vídeo.

Se ha probado a cambiar el parámetro en la URL, y volver a ejecutar, notablemente el vídeo tarda mucho menos en empezar a reproducirse, aproximadamente 15’’.
Esto pasa porque el parámetro que se cambia es solo el de texto, si el parámetro que se cambiase es el que hace que cambie el vídeo y no los textos de personalización, tarda un tiempo diferente.

Se supone que el servidor almacena los fragmentos personales del cliente, y solo una vez los fragmentos comunes a todos los clientes.

Conclusiones:

- Ventajas:
  - No es necesario tener control sobre el cliente para aplicar la personalización, todo se hace desde el servidor, lo que me permite tener un campo más amplio de dispositivos que pueden visualizar el vídeo.
  - No hay que almacenar un vídeo completo por cliente consumidor del vídeo, sino que solo se almacenan los fragmentos personalizados, los fragmentos comunes solo se almacenan una vez para todos los clientes.
- Inconvenientes:
  - Al pulsar play sobre el vídeo no se ve el vídeo de inmediato, lo que produce esperas por parte del cliente, y a su vez rechazo, muchos clientes acabarán por cerrar el navegador antes de que el vídeo se reproduzca.
  - Al renderizarse el vídeo en el servidor, la capacidad de proceso aumenta, y permitirá menos peticiones simultáneas.

b) **Manda imagen estática con enlace a LandingPage al email** (almacena un vídeo por cliente en el servidor) [3]

En esta opción, se rellena un formulario con parámetros de personalización como son: Nombre, apellido, nombre de la compañía y el email, todos estos parámetros de tipo texto, que después se añaden en el propio vídeo.

En el momento que enviar el formulario para la personalización del vídeo, aparece un mensaje indicando que tardará aproximadamente un minuto en mandarte un correo.
Dicho correo tendrá como contenido del mensaje una foto, la cual contiene un enlace a la página web que aparece el vídeo con tu personalización.

Lo que se supone que hace es: en el momento de mandar el vídeo al correo, edita el vídeo completo, por ello tarda un tiempo determinado y no es inmediato el envío.

El vídeo queda almacenado en el servidor durante un largo tiempo, probando semanas después, este sigue estando y reproduciéndose con los parámetros indicados guardando un vídeo por cada personalización diferentes que haya.

Conclusiones:

- Ventajas:
  - Al enviarse el enlace al email, el cliente consumidor invierte un tiempo en acceder al correo, tiempo que utiliza el servidor en personalizarlo.
  - El cliente no “percibe” las esperas ya que no está esperando sin hacer nada, sino que en el lapso de tiempo que accede al correo, el vídeo se personaliza.

- Inconvenientes:
  - Las esperas siguen siendo las mismas que en la opción anterior.
  - Se crea un vídeo entero e individual para cada consumidor, lo que aumenta el almacenamiento en el servidor.

3. **Personalización de vídeo por fragmentos** (solo guarda un vídeo en el servidor para todos los clientes) se personaliza en el momento en el que el cliente pulsa el play.

   El cliente que pulsa play, empieza a ver el vídeo al instante, sin sufrir esperas o retardos, mientras éste se está reproduciendo, el fragmento que se quiere personalizar está editándose de manera inmediata.

   Cuando el fragmento personalizado tiene que reproducirse ya ha sido editado mientras se reproducía el fragmento anterior, lo que hace una personalización en tiempo real.

   Hay que enviar el vídeo por fragmentos de manera que estos sean transparentes para el usuario, reproduciéndolo en el lado del cliente con un reproductor html5 como una “play list”

   Conclusiones:

   - Ventajas:
     - No almacena más que un vídeo, por lo que no necesita gran espacio de almacenamiento.
     - Si el vídeo no se llegara a ver entero, no se consumirían recursos innecesarios, ya que el proceso de personalización se pararía.
     - La personalización se hace a la vez que el envío del vídeo, por lo que tiene muy pocas esperas al reproducirse.

   - Inconvenientes:
     - El proceso de desarrollo de la aplicación es más complejo.
     - Se consumen recursos del servidor cada vez que se visualiza el vídeo.

2. Formatos de vídeo

Un contenedor contiene varios componentes del vídeo como las imágenes del vídeo, audio y metadatos. Entre los formatos de los contenedores más conocidos están OGG, 3GP, AVI, MOV, MP4, Flash Vídeo (FLV), WebM de Google, Matroska (MKV), entre otros.

Entre los códecs de vídeo, audio y metadatos tenemos:

- ➔ **Códec de vídeo:** H.263, H.264, H.265, VP8, MPEG-4, etc.
- ➔ **Códec de audio:** Advanced Audio Coding (AAC), MP3, Vortis, Windows Media, Audio (WMA), etc.
- ➔ **Metadatos:** XML, RDF, XMP, SRT, etc.

Cada archivo de vídeo contiene ciertos atributos como:

- ➔ **El tamaño del Marco (Frame size):** Es la resolución que tiene una imagen, cuanto mayor sea la resolución, mayor calidad tendrá el vídeo, pero también requerirá más ancho de banda para la transmisión.
- ➔ **La tasa del Marco (Frame rate):** Es la velocidad con que los marcos son capturados y posteriormente reproducidos. La tasa del marco más comunes son 15 frames per second (fps), 24 fps, 25 fps, 30 fps, y 60 fps.
- ➔ **Tasa de bit (Bitrate):** La tasa de bit es el número de bits que se transmiten en un tiempo determinado. El bitrate es una combinación entre el contenido de vídeo y del audio en el archivo. Normalmente un Bitrate (Tasa de bit) alto implica una mejora calidad del vídeo y un tamaño mayor.
- ➔ **Tasa de muestreo del audio:** El audio a su vez, también es comprimido usando los diferentes códecs. Dependiendo del códec que se utilice añadirá entre unos 128 - 256 kbps a la tasa de bit.

El formato utilizado para los vídeos utilizados en el proyecto es MP4 que es el contenedor digital de formato multimedia más utilizado para almacenar vídeo y audio, pero también permite almacenar otros datos como subtítulos y más.
Permite la transmisión Streaming.

La extensión mp4 es una abreviación de MPEG-4, el cual está comprimido y no contiene sólo vídeo. Si solo contiene audio el formato se expresa m4a.

El nombre de sus extensiones puede ser:
.mp4, .m4a, m4p, .m4b, m4r y m4v

El MIME utilizado por los navegadores para referirse a él es video/mp4 y sigue el estándar ISO/IEC 14496-14.

Sobre este formato será necesario conocer los metadatos que contiene, ya que con Ffmpeg tendremos que cambiar algunos de ellos para que en la personalización todo funcione correctamente.

3. Librerías / Herramientas de edición de vídeo

Se han buscado las librerías o herramientas que nos permiten hacer la personalización del vídeo mediante programación en GO.

Se han encontrado las siguientes:

1. **Ffmpeg:** es una colección de software libre, multiplataforma que sirve para tratar archivos multimedia que puede decodificar, codificar, transcodificar, multiplexar, demultiplexar, transmitir, filtrar y reproducir casi cualquier formato existente.
   La colección de software está compuesta por:

   - **Ffmpeg**: convertidor de audio y vídeo que puede convertir también incluso multimedia procedente de una fuente en directo.
   - **Ffserver**: servidor de streaming de audio y vídeo.
   - **Ffplay**: reproductor multimedia que usa las librerías ffmpeg y SDL.
   - **Ffperobe**: analizador de flujos multimedia.
   - **Libavutil**: librería que contiene un conjunto de funciones que permite simplificar la programación.
   - **Libavcodec**: contiene codificadores y decodificadores para gran cantidad de códecs de audio y vídeo.
   - **Libavformat**: contiene multiplexores y demultiplexores para los formatos de los contenedores multimedia.
   - **Libavdevice**: contiene dispositivos de entrada y salida que graban y renderizan los entornos de software de I/O multimedia más comunes. 8Video4Linux, VfW, ALSA.
   - **Libavfilter**: contiene filtros multimedia.
   - **Libsawcale**: hace operaciones de escalado de imagen y conversiones de color.
   - **Libswresample**: realiza operaciones de resampling, rematrixing y conversión de formato.

   Mas información en [4]

2. **OpenCV:** es una librería de tratamiento de imágenes, muy utilizada en la inteligencia artificial y en el reconocimiento de objetos en imágenes, que también nos permite editar nuestro vídeo.

   Es multiplataforma, se puede desde reconocimiento de objetos, hasta calibración de cámaras, visión etérea y visión robótica.

   Al ser una herramienta orientada más a otros aspectos que a la edición de vídeo, aunque se permita, es más conveniente Ffmpeg por sus amplias capacidades de edición.

4. Formas de enviar vídeo por fragmentos

Una vez se han encontrado las diferentes formas con las que hasta el momento se ha hecho la personalización de vídeo, como una de ellas es **enviar el vídeo por fragmentos,** y es la que nos interesa, se ha investigado las formas que hay de llevarlo a cabo.

POSIBLES FORMAS DE ENVIAR VÍDEO POR FRAGMENTOS

1. **Chunked transfer encoding:**(no válida) Envío de datos por fragmentos, mediante HTTP, que consiste en el envío de datos mediante “chunks” traducido al español, trozos o cachos (chunk: fragmento de datos a nivel de HTTP para enviar un recurso).

   Codificación utilizada para la transferencia de un archivo en fragmentos. Cada trozo está precedido de su tamaño y el cliente no necesita saber el tamaño completo del recurso sino el tamaño del fragmento actual.

   La transmisión finaliza cuando el cliente encuentra un fragmento con longitud 0. S**irve para la transferencia fragmentada con contenido generado dinámicamente.**

   Si se especifica en la cabecera HTTP un campo Transfer-Encoding=” chunked”, el cuerpo del mensaje consta de un número no especificado de trozos.

   Hay **límite de datos (bytes) necesarios que un navegador necesita** para comenzar a reproducir un vídeo en fragmentos.

   Cuanto más chunking más sobrecarga en la producción y análisis de los trozos ya que chunking agrega complejidad al cliente. Si el envío del vídeo es pequeño (archivo poco pesado) no es conveniente hacer con chunking ya que le puede costar más que enviar el vídeo completo directamente.

   Existe la posibilidad de enviar los datos fragmentados **comprimidos**, gzip soporta la comprensión de fragmentos.

   Conclusiones:

   - Ventajas:
     - No se necesitan herramientas externas al lenguaje de programación utilizado, ya que todo se hace con el control de cabeceras.
   - Inconvenientes:
     - Se necesita saber con exactitud cómo está codificado cada vídeo, y conocer muy bien el formato que lo encapsula, opción muy difícil de llevar a cabo.

2. **VÍDEO DIVIDIDO EN VÍDEOS:** Envío de vídeos divididos en fragmentos, la idea de esta opción consiste en dividir el vídeo que se desea personalizar en diferentes fragmentos y que cada uno de los fragmentos sea un vídeo independiente, ocupando un lapso de tiempo que se desee, para personalizar sólo el archivo de vídeo que se quiere personalizar, en lugar de todo el vídeo.

   En el lado del cliente se crea un reproductor escrito en html5 para reproducir una lista de vídeos (“PlayList”), mediante JavaScript.

   Conclusiones:

   - Ventajas:
     - Opción sencilla de implementar
   - Inconvenientes:
     - En la barra del reproductor donde indica el tiempo de reproducción no aparece como si de un solo archivo de vídeo se tratara.
     - Todos los vídeos deben estar hechos antes de la reproducción, por lo que no se personaliza en tiempo real.
     - Se necesita gran espacio de almacenamiento en el servidor ya que cada fragmento personalizado hay que almacenarlo.

3. Transmisión por streaming: se explica a continuación las diferentes formas de mandar vídeos por streaming.

5. Vídeo Streaming

El streaming consiste en la distribución de contenido multimedia (vídeo o audio) a través de una red de ordenadores de forma continua, de modo que el usuario lo ve a la vez que se descarga.

Funciona a través de un buffer de datos que almacena el flujo de descarga en el equipo del usuario para mostrárselo de inmediato.

La retransmisión (corriente que fluye sin interrupción, normalmente la difusión de audio o vídeo) requiere una conexión con un ancho de banda mayor o igual que la tasa de transmisión del servicio.
A continuación, se explican los protocolos existentes para realizar streaming y los tipos de streaming que hay.

Protocolos

La información se transmite desde el servidor al cliente utilizando un protocolo de transporte diferenciados claramente en dos categorías:

- **Protocolos no basados en HTTP y Protocolos basados en HTTP**

Protocolos no basados en HTTP:

- **Real-Time Transport Protocol (RTP):** protocolo de nivel de aplicación para la transmisión en tiempo real. Funciona sobre UDP, por lo que no garantiza que los paquetes lleguen a su destino. Se considera el principal estándar de transporte de audio y vídeo en las redes IP.

  Funciona en unicast y multicast.
  - Unicast: unidifusión o difusión única. Envío de información de un único emisor a un único receptor.
  - Multicast: multidifusión. Envío de un único emisor a varios receptores.

  Este protocolo multiplexa el vídeo y audio y lo manda a través de la red en un único flujo de paquetes UDP.

  Cuando el usuario los recibe el reproductor va desensamblando los paquetes y los va añadiendo en un buffer, hasta que está suficientemente lleno para empezar la reproducción.

- **Real Time Streaming Protocol (RTSP):** protocolo de nivel de aplicación cuya función es establecer y controlar las sesiones multimedia entre servidor y cliente que permite mantener una o varias sesiones.
  Es un tipo de control remoto en red a un servidor multimedia.
  No depende de otro protocolo de transporte para realizar control de las sesiones.

- **RTP Control Protocol (RTCP):** protocolo que proporciona información de control para el protocolo RTP. Su función principal es informar de la calidad del servicio proporcionado por RTP.

- **Real-Time Messaging Protocol (RTMP):** protocolo propietario de Adobe.
  Pensado para la transmisión de streams de vídeo, audio y otros datos.
  Tiene baja latencia y es usado por Flash Player para reproducir contenido bajo demanda y en directo.
  Los navegadores necesitan un plugin para poder usarlo.
  Trabaja con el protocolo TCP.

- **File delivery over Unidirectional Transport Protocol (FLUTE):** protocolo utilizado para la entrega de archivos sobre enlaces unidireccionales usando UDP.
  Puede ser unicast o multicast.
  Permite trabajar en muchos tipos de redes, LANs, WANs, intranets…
  Proporciona fiabilidad en la transmisión, muy necesario en multicast.

Pero estos protocolos existentes tienen algunos inconvenientes como:

- Hay gran cantidad de redes de distribución de contenidos, de las cuales muchas no soportan RTP.

- Los firewalls tampoco admiten este tipo de paquetes.

- En RTP y RTMP se tiene que gestionar una sesión diferente para cada cliente.

- También, en algunos de ellos necesitan utilizarse junto a otros, con lo que incrementa la información que se transmite por la red.

Por ello se ha empezado a utilizar en gran medida otros protocolos que utilizan HTTP para la transmisión de datos **(HTTP Streaming).**

Protocolos basados en HTTP:

- **HTTP Live Streaming (HLS) de Apple:** su aplicación está enfocada a la comunicación de los dispositivos con sistemas operativos iOS y OS X.
  Utiliza igual que el resto de protocolos basados en HTTP chunks para transportar el vídeo, en este caso con la extensión ".ts". Y dispone de un archivo Manifest llamado M3U8 que contiene la información necesaria para reproducir los diferentes chunks disponibles.

- **Microsoft Smooth Streaming (HSS):** la idea es igual que el anterior con algunas diferencias notables.
  Los fragmentos son de 2 segundos.
  Tiene también un archivo Manifest que en este caso pueden ser de dos tipos:
    Manifest en MSS (extensión .ism)
    Manifest MSS del servidor web (extensión .ismc)
  Necesita el reproductor de Microsoft en el lado del cliente llamado Silverlight.

- **Adobe HTTP Dynamic Streaming (HDS):** estándar de transmisión adaptativa que permite la distribución de vídeos en directo y bajo demanda a velocidades de bits adaptables. Es similar a los anteriores, pero esta vez desarrollado por Adobe, y con ligeras diferencias.
  Algunas diferencias son:
  - Combina los métodos de descarga progresiva y las de streaming.
  - Extensión de los fragmentos .F4F
  - El archivo Manifest (extensión .F4M)
  - Para reproducir su contenido en el lado del cliente el navegador necesita un plugin.

- **Dynamic Adaptive Streaming over HTTP (DASH):** Opción elegida para llevar a cabo el envío de vídeo del proyecto que se explica un poco más adelante.
  Es un estándar que se ha desarrollado para evitar los problemas de los protocolos anteriores, ya que han sido creados por una empresa privada y solo admiten sus propias tecnologías.

Una tabla de clasificación de los protocolos podría ser:

Protocolos no basados en HTTP	Protocolos basados en HTTP
Real-Time Protocolo (RTP)	HTTP Live Streaming de Apple (HLS)
Real-Time Streaming Protocol (RTSP)	Microsoft Smooth Streaming (MSS)
RTP Control Protocol (RTCP)	HTTP Dynamic Streaming de Adobe (HDS)
Real-Time Messaging Protocolo (RTMP)	Dynamic Adaptative Streaming over HTTP (DASH)
File delivery over Unidirectional Transport Protocol (FLUTE)	

Tabla3. Protocolos HTTP y no HTTP

Protocolos clasificados por estrategia de distribución de contenido entre los nodos de la red:

- **Estrategia PUSH:** tras establecer conexión entre el cliente y servidor, el servidor transmite los paquetes hacia el cliente, hasta que éste termina la conexión. Es el servidor el que mantiene el estado de la sesión con el cliente.

- **Estrategia PULL:** ahora el cliente es quien solicita el contenido al servidor, es decir, el cliente es quien inicia la comunicación.
  El método de descarga progresiva utiliza esta estrategia. En descarga progresiva, cuando un cliente realiza una solicitud HTTP al servidor e inicia la descarga los más rápido posible. Cuando el cliente tiene el buffer lo suficientemente lleno, iniciará la reproducción mientras el contenido se sigue descargando y llenando el buffer.

Tipos de streaming

- **Streaming tradicional:** se emplea un servidor multimedia para transmitir el contenido.
  Los protocolos que se emplean en este tipo de streaming son: RTSP, RTMP y RDT.
  Para explicar este tipo, me basaré en el protocolo RTSP (Real-Time Streaming Protocol), que es un protocolo con estado, lo que indica que desde que el cliente establece la conexión con el servidor, éste hace un seguimiento de la misma hasta que se finaliza la conexión.

  Durante la conexión, el servidor envía el paquete (RTP) que contiene unos 11 milisegundos de vídeo.

  Los paquetes se transmiten sobre UDP o TCP, lo que puede provocar problemas con los firewalls o proxis en caso de UDP o retardos en caso de TCP.

  Este tipo se diferencia del resto por:

  - El servidor envía paquetes con el bit-rate que se ha codificado el vídeo y no cambia.
  - El servidor envía paquetes hasta que se llena el buffer del cliente, lo que conlleva que el bit-rate de codificación del vídeo debe ser menor que el ancho de banda disponible del usuario, si no queremos que la reproducción tenga cortes continuos.

- **Descarga progresiva:** el contenido se introduce en un servidor web normal, que sirve páginas web. El cliente necesita saber la URL del archivo multimedia para poder descargarlo.
  Si paramos la reproducción de una descarga progresiva y esperamos a que se complete la descarga, después podremos ver el vídeo completo sin cortes, incluso sin conexión a Internet.
  Al principio, cuando empezó esta tecnología, había que esperar a que se descargara el archivo para empezar a reproducirlo, pero ha evolucionado y ahora es capaz de empezar a reproducir el archivo mientras se está descargando.

  Pero la descarga progresiva tiene algunos inconvenientes:

  - Consume un gran ancho de banda.
  - Si el buffer se ha cargado, y por cualquier motivo deja de visualizarse el contenido, el resto del buffer cargado y no visualizado se habrá perdido, y se habrá consumido un ancho de banda innecesario.
  - No se puede cambiar la calidad del vídeo una vez se ha comenzado a transmitir.

  Por estas desventajas surge el tercer tipo de streaming.

- **Streaming adaptativo:** la principal diferencia entre este tipo y el anterior, es que en este caso se realizan pequeñas descargas progresivas, en vez de realizar una única descarga de todo el fichero, el contenido de vídeo y audio se trocea en pequeños fragmentos (chunks) y se codifica según el formato en el que se quiera transmitir.
  Cada fragmento es independiente del anterior, por lo que pueden ser decodificados de forma independiente.

  La principal ventaja de este método es que se pueden generar diferentes segmentos de diferentes calidades y el cliente, dependiendo de su ancho de banda o del dispositivo utilizado, puede elegir entre una calidad y otra.

  Este tipo de streaming tiene varias implementaciones existentes como **HTTP Live Streaming (HLS) de Apple, Microsoft Smooth Streaming (HSS), Adobe HTTP Dynamic Streaming (HDS)**

CONCLUSIONES:

Los tres primeros protocolos de streaming adaptativo tienen una serie de restricciones, cada uno utiliza diferentes formatos en sus segmentos y sus ficheros (manifest) de descripción de datos. Por lo que, si un dispositivo quiere reproducir contenido de un servidor de cada plataforma, deberá soportar el correspondiente protocolo, que además son propietarios.
Por lo que se ha decidido utilizar el protocolo Dynamic Adaptive Streaming over HTTP (DASH) que explicamos a continuación:

Diferencias de los protocolos anteriores:

Protocolo	Año	Contenedores	Manifest	Códecs Vídeo	Códecs Audio
RTMP (Adobe)	2009	MP4, FLV	SMIL	H.263, H.264	AAC, MP3
HLS (Apple)	2009	MPEG-TS,	M3U8	H.263, H.264	AAC, AC3, MP3
MMS (Microsoft)	2009	MP4	ISM	VC-1, H.264	WMA, ACC
HDS (Adobe)	2010	MP4,FLV	F4M, SMIL	H.264, VP-6	ACC, MP3
MPEG-DASH (MPEG)	2012	MP4,MPEG-TS	MPD	Varios, HEVC	Varios

Tabla4. Protocolos Streaming

EL PROTOCOLO DASH

Como es el protocolo elegido para desarrollar el proyecto se va a profundizar un poco en él explicando algunos de los aspectos más importantes de éste:

Dynamic Adaptive Streaming over HTTP (DASH ó DASH-MPEG) es un estándar publicado como ISO/IEC 23009-1:2012, utilizado para enviar contenido por la web por streaming adaptativo sobre HTTP que ha sido desarrollado por MPEG (Moving Picture Expert Group). Se convirtió en un estándar internacional en noviembre de 2011.
Supone una gran ventaja respecto a los anteriores protocolos definidos que ya que no opera bajo ningún estándar propietario.

DASH define dos formatos:

- **Media Presentation Description (MPD):** consiste en un fichero de metadatos escrito en XML que describe el contenido que se envía.
  Compuesto por las distintas versiones del contenido multimedia con su descripción (diferentes alternativas), direcciones URL y otras características.
- **Segmentos:** contiene las secuencias de contenido multimedia en forma de trozos, en uno o varios archivos.
  Pueden ser de duración variable y deben estar disponibles cuando el cliente los solicite.
  Para los servicios en vivo, sin embargo, estarán disponibles mientras se va produciendo el contenido.

Los datos son contenidos en el servidor, y pasarán al cliente mediante este protocolo, para ello se utilizara la estrategia de distribución PULL lo que indica que el cliente es el que pide los fragmentos al servidor, sabrá que fragmento tiene que pedir gracias al fichero MPD (Media Presentation Description).

Algunas de las **características** que define este protocolo son:

- Soporta dos tipos de streaming:
  - **Streaming bajo demanda (VoD):** este tipo de streaming bajo demanda permite a los usuarios elegir el contenido que desean reproducir. El usuario puede interactuar mientras el contenido se reproduce, puede pausarlo, ir hacia adelante, ir hacia atrás en cualquier momento.
    La transmisión será unicast.
  - **Streaming en vivo:** El contenido debe estar disponible para todos los dispositivos que soliciten el servicio. No existe interactividad con los usuarios, solo pueden parar la transmisión.
    La transmisión será multicast.

- Posibilidad de formato
  - **MPEG2-TS:** estándar diseñado para la comprensión de señales de vídeo que necesita gran ancho de banda.
  - **MPEG-4:** Mejora el rendimiento de MPEG-2 pero para flujos más bajos, pero es más eficiente y mejor que MPEG2-TS.

- Utiliza HTTP.

- Control total de streaming por parte del cliente.

- **Soporta cambios de calidad sin cortes:** detecta el ancho de banda del cliente y los recursos computacionales y sirve fragmentos contenidos codificados en el bitrate más adecuado para la mejor experiencia visual y sin cortes.

- **Admite segmentos con duración variable.**

- Soporta múltiples sistemas de protección de contenido.

El estándar tiene varias **ventajas**:

- Funciona independientemente del códec, por lo que emplea contenido multimedia H.264, WebM y MPEG2TS. Esto quiere decir que funciona para cualquier especificación de las anteriormente descritas (HLS, HSS y HSD).

- Permite audio en diferentes idiomas y seleccionar el más conveniente.

- Permite emplear segmentos de longitud variable, a diferencia de los anteriormente descritos, que tenían longitudes fijas, y cada uno de ellos, una longitud diferente.

- El mismo contenido disponible en diferentes URLs, por lo que puede contenerse el vídeo en varios servidores, obteniendo así un mayor rendimiento.

El encargado de reproducir el contenido multimedia es el cliente(consumidor) y lo hará mediante el siguiente proceso:

1. **Obtiene el MPD** mediante petición GET de HTTP, correo electrónico u otro método para transporte de ficheros.

2. **Lo parsea** para obtener la información necesaria para la reproducción del elemento multimedia.

3. Una vez se tiene la información, selecciona las características necesarias y comienza **obtener** los **diferentes segmentos** mediante peticiones HTTP por streaming.

ARQUITECTURA

MPEG-DASH define los siguientes conceptos:

- Fichero **MPD (Media Presentation Description):** hay dos opciones:
    - (SMPD): Static Media Presentation Data para contenido bajo demanda.
    - (DMPD): Dynamic Media Presentation Data para contenido en directo.

- Proceso de creación de los MPD:
    - Se selecciona el vídeo que se quiere transmitir.
    - Se generan los vídeos en diferentes calidades o los diferentes fragmentos del vídeo
    - Se genera el MPD a partir de esos vídeos.

- El fichero de presentación de datos puede contener algunas de las siguientes partes:
    - Periodos (**Periods**): conjunto de las versiones codificadas del contenido multimedia. Cada uno puede tener un Set de Adaptación distinto.

      El Periodo tiene un tiempo de inicio (PeriodStart) que se determina a partir del atributo @start y en su ausencia se calculará haciendo la suma del PeriodStart anterior y el valor que contenga el atributo @duration.

    - Sets de adaptación (**Adaptation Sets**): cada Periodo tiene uno o varios de este tipo, cada Set de Adaptación proporciona información sobre uno o más componentes multimedia y las alternativas de codificación.

      Cada Set de Adaptación consta de una o más Representaciones, todas las ellas contienen el mismo contenido multimedia.

    - Representaciones (**Representations**): es una alternativa de codificación de un componente multimedia, que su diferencia con el resto de Representaciones por su bitrate, resolución, úumero de canales …
      Cada Representación contiene uno o más Segmentos y comienza al inicio del Periodo y acaba al final del mismo.

    - Segmentos (**Segments**): cada una de las partes en las que se ha dividido un stream multimedia.

- Subsegmentos **(Subsegments)**: En caso de que los segmentos estén divididos en subsegmentos, esos están presentes en un Índice de Segmentos, proporcionando el índice de tiempo y el rango de bytes de cada subsegmento.

Se puede indicar donde se encuentra cada uno de los segmentos mediante una url que ira dentro de una de estas tres etiquetas:

- **SegmentList**: uno o más elementos SegmentList, dónde cada uno de ellos puede contener varios SegmentURL, atributo que contiene la dirección URL donde se encuentran los segmentos multimedia.

- **SegmentTemplate**: este elemento utiliza identificadores que se le asignan valores dinámicamente de una lista de segmentos.

- **SegmentBase**: suficiente en caso de haber un único segmento multimedia en la Representación, en caso de haber varios se utilizará segmentList o SegmentTemplate.

Hay 4 tipos de segmentos:

- **Initialization Segment:** contiene la información necesaria de inicialización perteneciente a los segmentos de la representación para poder reproducirlos correctamente.

- **Media Segment:** encapsula los streams multimedia y más información necesaria para poder reproducir esos streams correctamente.

- **Index Segment:** contiene información sobre la indexación para uno o más Media Segments.

- **Bitstream Switching Segment:** contiene la información necesaria para cambiar a la Representación a la que pertenece.

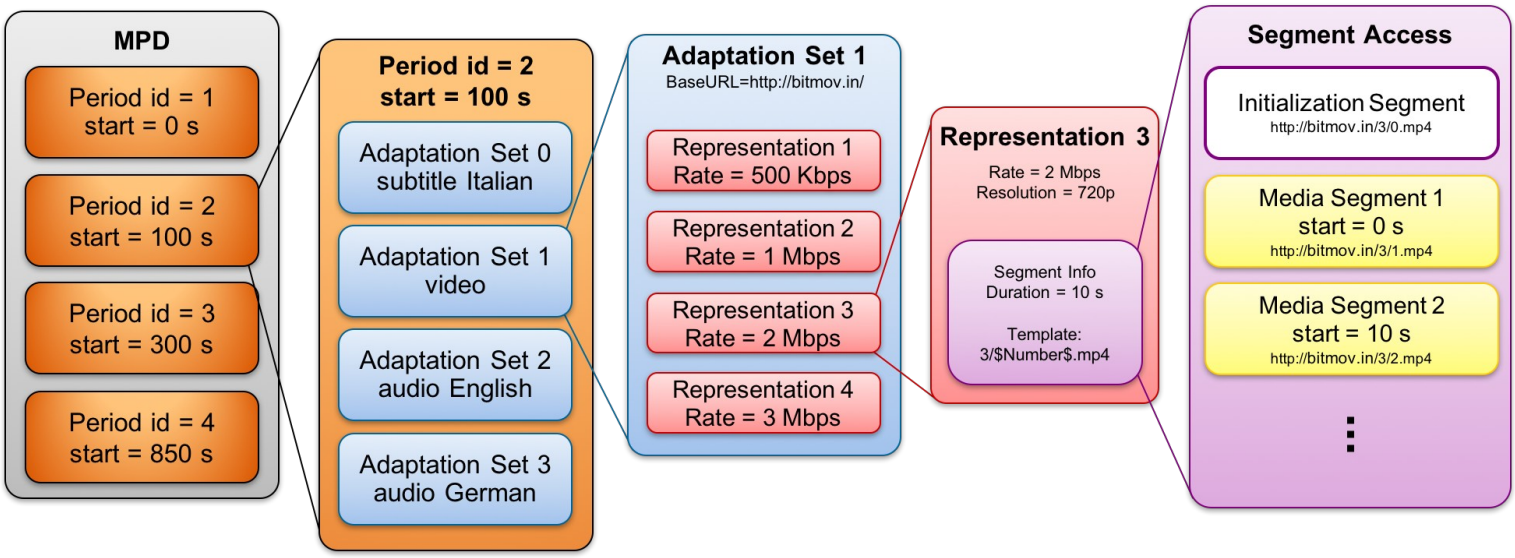


Figura2. slideshare.net/christian.timmerer/mpegdash-overview-stateoftheart-and-future-roadmap

➢ Capítulo 3. Implementación

Se ha llevado a cabo la implementación con un conjunto de herramientas principales como son GPAC (MP4Box) y Ffmpeg, combinando sus funcionalidades se ha personalizado el vídeo mientras se están reproduciendo los primeros fragmentos del mismo con el protocolo DASH.

A través del protocolo DASH se comienza a reproducir los primeros fragmentos del vídeo, mientras tanto se edita el vídeo, para a continuación con la herramienta MP4Box generar nuevos fragmentos del vídeo editado, tras editar el vídeo se sustituyen los fragmentos editados por nos lo editados, por último, cuando el vídeo llegue a la parte de la personalización, los fragmentos que reproducirá serán los últimos generados a partir del vídeo editado.

1. Lenguajes usados

Se utiliza el lenguaje de programación GO, se ha optado por este lenguaje ya que en la empresa en la que se desarrolla el proyecto, se trabaja con dicho lenguaje para programar la parte funcional de la aplicación, por lo que, si se fuera a reutilizar la aplicación desarrollada en el proyecto, se escribe en el mismo lenguaje por facilidad de integración.

Para poder utilizar correctamente este lenguaje, se ha necesitado un periodo de previo aprendizaje.

2. Tecnologías usadas

- **Firebug**: es una extensión de Firefox diseñada en especial para desarrolladores y programadores, En el caso del desarrollo de este proyecto se ha utilizado para ver los paquetes http que se envían durante la transmisión del vídeo.

  En ella se pueden ver diferentes aspectos de la web, dependiendo de las necesidades de cada desarrollador.
  Principalmente su uso es para:

  a) Inspeccionar HTML y modificar el estilo y el diseño en tiempo real.

  b) Usar el depurador para JavaScript

c) **Analizar con precisión y el rendimiento de la red:** esta es especialmente la opción que interesa para este proyecto. En este tipo de uso se puede:

   - Examinar cabeceras HTTP
   - Monitorizar paquetes de tipo XMLHttpRequest (paquetes que envía DASH)
   - Tiempos que tarda en recibir cada paquete.

- **Ffmpeg:** herramienta de línea de comandos utilizada para la personalización del vídeo, que se ha explicado ya anteriormente en esta memoria.

- **MP4Box**(GPAC)**:** es un conversor MPEG-4, puede importar vídeo MPEG-4 y flujos de audio en un contenedor MP4.

  Herramienta que se usa en la línea de comandos utilizada para la fragmentación del vídeo para que DASH pueda interpretarlos, además también genera el archivo Manifest de DASH.

- **Eclipse**: entorno de desarrollo para llevar a cabo el proyecto, después se explica la configuración de éste para el proyecto.

  Se ha necesitado instalar diferentes plugin para poder trabajar con el entorno de desarrollo con todas las funcionalidades necesarias para acoplarlo con GO.

  Los plugin son los siguientes:

  - **goclipse**: para poder trabajar con GO desde el entorno de desarrollo Eclipse.

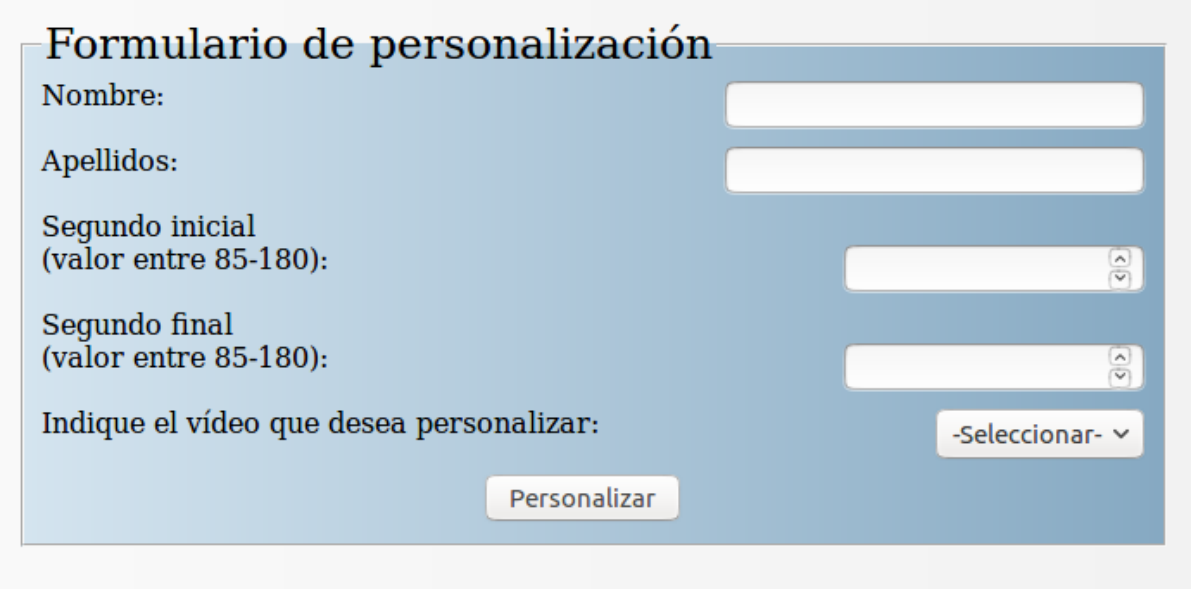
  se necesita instalar las herramientas de go: golang-golang-x-tools

  - godef: muestra la definición de métodos y propiedades del código fuente GO.
  - godoc: documentación de go
  - gofmt: herramienta para formatear automáticamente el código fuente de GO.
  - librería Ffmpeg: para poder utilizar Ffmpeg en go.
  - Gocode: sirve para el auto completado del lenguaje GO.

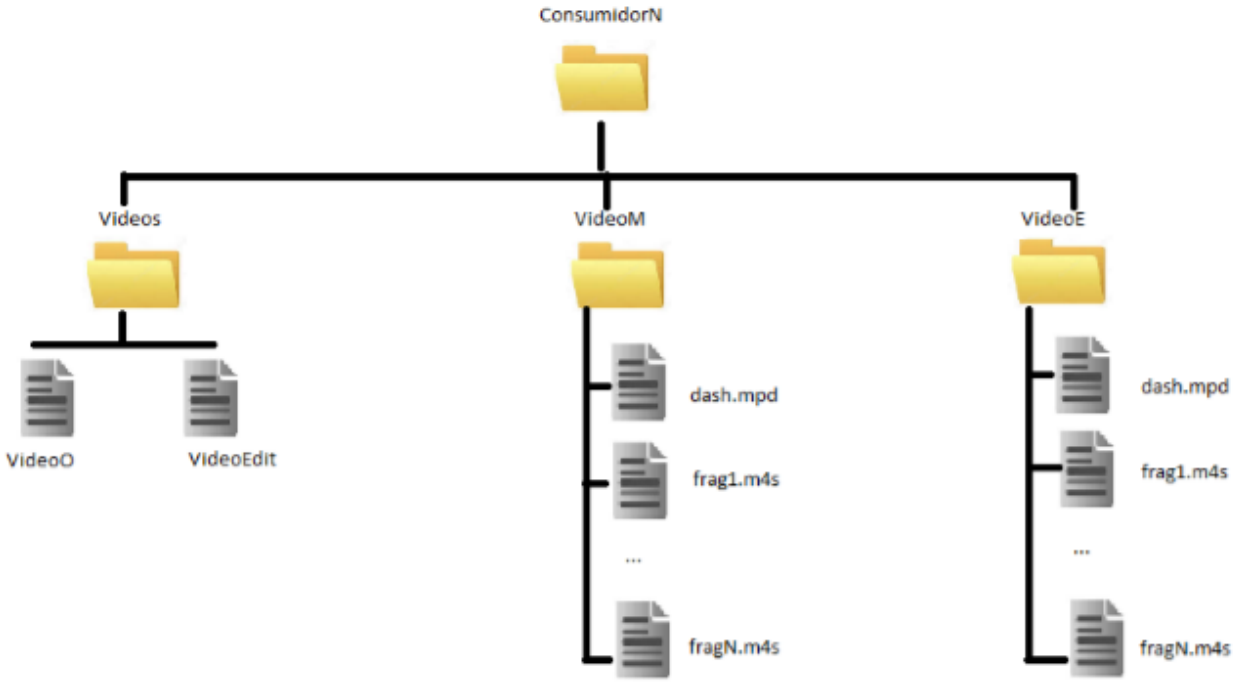
- **MediaInfo:** herramienta de consola de comandos Linux utilizada para ver los metadatos de los elementos multimedia.

3. Implementación utilizada.

1. En primer lugar, el cliente consumidor del vídeo, a través de una página web, envía un formulario con los parámetros de personalización del vídeo. El formulario es el siguiente:



2. Cuando el consumidor envía el formulario, la aplicación automáticamente genera una estructura de directorios temporal, para cada consumidor que desee personalizar su vídeo.



3. Tras tener la estructura de directorios creada, se copia el vídeo original correspondiente que se haya seleccionado, que se desea mostrar y permanece alojado en el servidor, al directorio Vídeos y se hace la fragmentación y generación del vídeo con el siguiente método.

```
func (this *Ffmpeg) FragVideo(fileIn string, pathOut string) error {

    parameters := ff.NewParamSet(
            ff.NewParam(nil, "y"),
            ff.NewParam("out", pathOut),
            ff.NewParam("dash", 3000),
            ff.NewParam("segment-name", "frag"),
    )

    input := ff.NewOutput(fileIn, parameters)
    err := this.LaunchCommand("MP4Box", input)

    return err
}
```

Esta función se traduce en ejecutar el siguiente comando en la terminal de Linux:

MP4Box -y -out /home/viwom.inet/airuzubieta/GoogleDrive/TFG/Proyecto/videos/Aida/VideoE/dash_video.mpd -dash 3000 -segment-name frag /home/viwom.inet/airuzubieta/GoogleDrive/TFG/Proyecto/videos/Aida/Videos/VideoEdit.mp4]

Explicación del comando:
Se utiliza la herramienta MP4Box de GPAC la cual tiene los siguientes parámetros:

- **y:** indica que, si los fragmentos y mpd que se desean generar, existen actualmente, que los sobrescriba, en caso de no indicar este parámetro, la aplicación dará error.

- **out:** se indica la ruta, junto con el nombre del fichero *.mpd donde se desea que se alojen los fragmentos resultantes del comando.

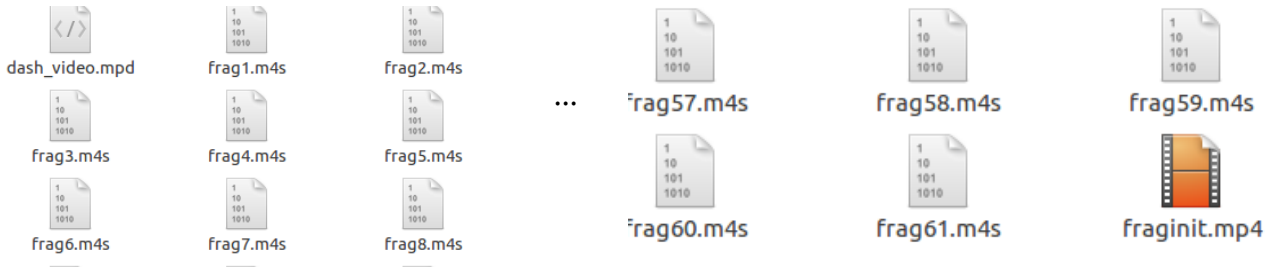
- **dash:** se indica el tamaño en milisegundos de cada fragmento.

- **segment-name:** se indica el nombre que tendrá cada fragmento añadiendo el número del mismo, por orden ascendente.

y por último se indica el nombre del vídeo el cual se desea fragmentar y generar el correspondiente MPD, con las características indicadas en los parámetros anteriores y conteniendo los metadatos del vídeo.

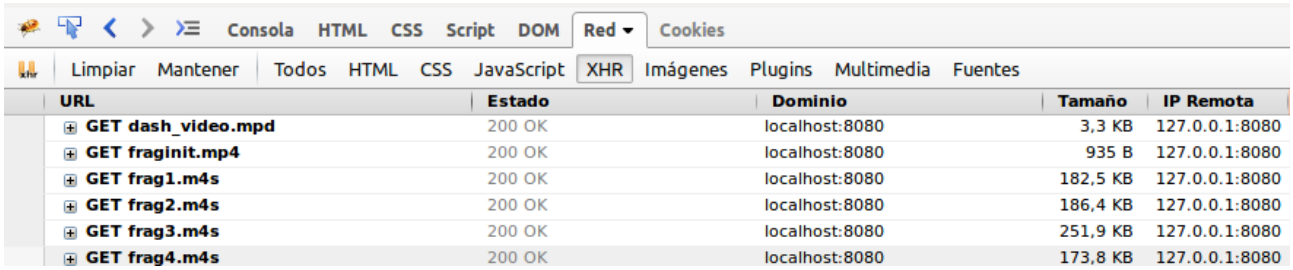
4. Una vez se ha copiado el vídeo, se hace la fragmentación del mismo y la generación de su fichero Manifest (MPD) con la herramienta MP4Box de GPAC, alojando los fragmentos y el MPD en el directorio VideoM, que es el directorio donde el cliente pedirá los fragmentos de vídeo.

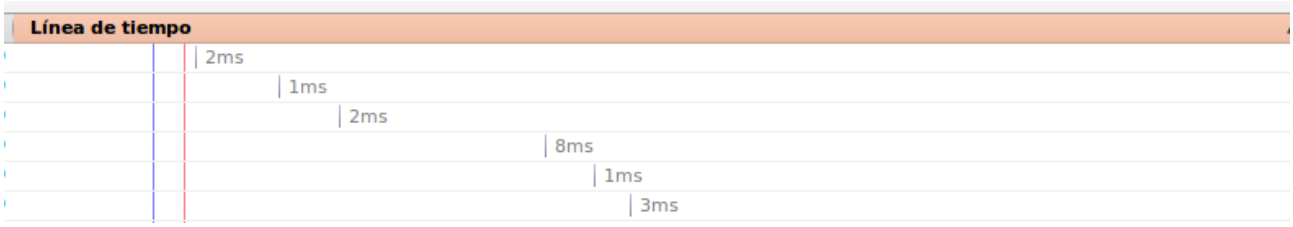
   Tras generarse el fichero MPD que es el primero, el fragmento inicial que es el ultimo y los correspondientes fragmentos nos queda algo como esto:



5. Una vez creada la estructura de directorios temporal y generados los fragmentos del vídeo original, se redirige a una página donde empieza a mostrarse el vídeo original en un reproductor creado con HTML5 y la librería dash.all.min.js que se ha utilizado para el correcto funcionamiento del reproductor con el protocolo de streaming utilizado DASH.

   Se ve la forma en la que el consumidor recibe el vídeo con la herramienta de Firebug instalada en el navegador, que muestra los fragmentos que va cargando en buffer.





6. Hay que tener en cuenta que el tamaño del Buffer del vídeo que se está reproduciendo es de 30 segundos, y según se va visualizando el contenido, se va cargando el buffer.
   Nombraremos más adelante la importancia de este aspecto a la hora de personalizar el vídeo de la forma implementada.

7. Mientras el vídeo se está reproduciendo y cargando el buffer, mediante la programación de un hilo en GO, se comienza a editar (personalizar) el vídeo, añadiendo los parámetros Nombre y Apellidos en el interior del vídeo, en el tiempo indicado en el formulario, con la herramienta anteriormente explicada Ffmpeg.

```
func (this *Ffmpeg) AddTextVideo(fileIn string, fileOut string, nombre string,
    apellido string, tini string, tfin string) error {

        valor := "drawtext= fontfile=/usr/share/fonts/truetype/ubuntu-font-family/Ubuntu-BI.ttf: text='"+
        nombre+" "+apellido+"': fontcolor=#ae2609: fontsize=30: box=1: boxcolor=black@0.0: \\"+
        "boxborderw=20: x=430:y=260:enable='between(t,"+tini+","+tfin+")'"

        parameters := ff.NewParamSet(
                ff.NewParam("i", fileIn),
                ff.NewParam("y", nil),
                ff.NewParam("r", 30),
                ff.NewParam("b", "500k"),
                ff.NewParam("vf", valor),
        )
        input := ff.NewOutput(fileOut, parameters)
        err := this.LaunchCommand("ffmpeg", input)

        return err
}
```

Esta función se traduce en ejecutar el siguiente comando en la terminal de Linux:

ffmpeg -i /home/viwom.inet/airuzubieta/GoogleDrive/TFG/Proyecto/videos/Parametro/Videos/Video.mp4 -y -r 30 -b 500k -vf drawtext=”fontfile=/usr/share/fonts/truetype/ubuntu-font-family/Ubuntu-BI.ttf:text='Parametro Prueba': fontcolor=#ae2609: fontsize=30: box=1: boxcolor=black@0.0: \ boxborderw=20: x=430:y=260: enable='between(t,90,100)'” /home/viwom.inet/airuzubieta/GoogleDrive/TFG/Proyecto/videos/Parametro/Videos/VideoEdit.mp4]

Explicación del comando:

Se utiliza la herramienta Ffmpeg, que previamente ha de estar instalada en el sistema en el que se está ejecutando la aplicación, es decir, en el servidor en el que se despliega.

Después se utilizan los siguientes parámetros para que añada texto en un vídeo determinado.

- i: indica la ruta y el nombre del vídeo que se desea editar

-y: indica a la herramienta que en caso de que el vídeo de salida ya exista, lo reemplace, ya que si no se pone este parámetro y el vídeo ya existe, la aplicación dará error.

-r: modifica ellos fotogramas por segundo del vídeo

-b: cambia el bitrate del vídeo (calidad del mismo)

-vf: añade un parámetro al vídeo, en nuestro caso de texto como lo siguiente:

drawtext=”fontfile=/usr/share/fonts/truetype/ubuntu-font-family/Ubuntu-BI.ttf:text='Parametro Prueba': fontcolor=#ae2609: fontsize=30: box=1: boxcolor=black@0.0: \ boxborderw=20: x=430:y=260: enable='between(t,90,100)'”

dónde cada parte de este comando es:

- fontfile: indica donde se encuentra el fichero de la fuente para poner el tipo de letra al texto
- text: el contenido que aparecerá en el vídeo.
-fontcolor: el color de la letra.
-fontsize: el tamaño de la letra.
-box: es el contenedor para alojar el texto.
-boxcolor: el color del contenedor y el grosor.
-boxborder: el borde.
-x: el valor de la coordenada x dónde aparecerá el texto.
-y:el valor de la coordenada y dónde aparecerá el texto.
-enable: sirve para indicar entre que segundos del vídeo se desea que aparezca el texto, en este caso se pasa como parámetro recogido del formulario.

y por último se pone la ruta y nombre del vídeo nuevo que se va a generar

8. Hay que tener en cuenta que los métodos que se muestran llaman siempre a la función LaunchCommand que es la que realmente ejecuta el comando en la terminal y será así:

```
func (this *Ffmpeg) LaunchCommand(command string, input ff.File) error {

    cmdline, err := ff.NewCommand(command, input)

    if err != nil {
        log.Println("[ffmpeg.go] Error generate cmdline "+command, err)
        return err
    }

    //Remove the default parameter "-y"
    commd := cmdline.Slice()[:len(cmdline.Slice())-1]

    var out []byte
    out, err = exec.Command(cmdline.Path, commd...).CombinedOutput()
    log.Println("[ffmpeg.go] Comando Calling ", cmdline.Path, commd)
    if err != nil {
        log.Println("[ffmpeg.go] Error Calling ", cmdline.Path, commd)
        log.Println(string(out))
    }
    return err
}
```

9. La edición del vídeo se hace sobre el vídeo original sin fragmentar, esto se hace así porque no se ha encontrado ninguna forma de editar los fragmentos que se envían, que tienen formato *.m4s y no contienen apenas metadatos del vídeo para poder editarse con una herramienta ya existente.

10. Hay que tener en cuenta, que el vídeo original que se tiene o que nos suben, tiene unos metadatos concretos y Ffmpeg al editar el vídeo para poner texto cambia los metadatos, por lo que a cada vídeo subido se ejecutara bajo el mismo comando que luego ejecutará la aplicación con la diferencia que no se añade nada, o sea, una línea en blanco, que a la larga en el vídeo no se notará.

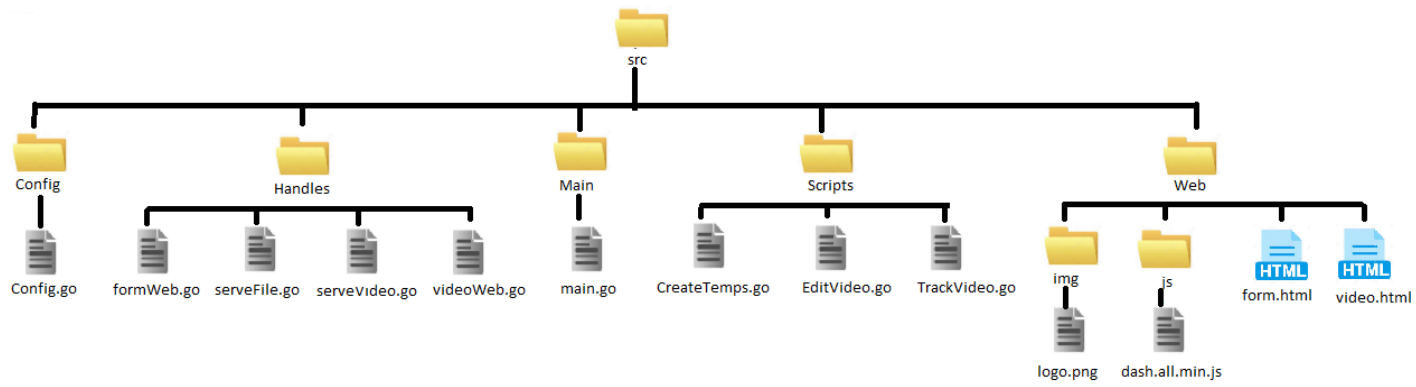
11. Una vez se ha editado el vídeo original, que tarda aproximadamente 60 segundos, un vídeo de 3 minutos, pero estos tiempos tienen una justificación y pueden variar.
    Posteriormente se explicará la importancia del tiempo que le cuesta editar el vídeo, y los parámetros de los que depende que le cueste más o menos tiempo ser editado.

12. Tras la edición, se fragmenta el vídeo editado, de la misma forma que se edita al principio de la aplicación el vídeo original para poder empezar a reproducir el vídeo, los fragmentos y el fichero mpd se alojan en el directorio creado anteriormente VideoE.
    La fragmentación debe ser exactamente igual en el punto 3 que en este, es decir, los fragmentos deberán tener el mismo tiempo que los que actualmente se están reproduciendo.

13. Una vez se ha editado y fragmentado el vídeo, se sustituyen todos los fragmentos del vídeo que se había empezado a reproducir alojados en el directorio VideoM, por todos los fragmentos que se han generado a partir del vídeo editado, que están alojados en el directorio VideoE, a excepción del fichero MPD, que se dejará el antiguo, que es el que actualmente se está leyendo por el cliente consumidor.

14. Por último, en el lapso indicado al rellenar el formulario, se pueden ver los parámetros introducidos en el vídeo en su reproducción, sin que el consumidor haya notado la edición del vídeo.

15. La estructura del programa del proyecto será algo como:



ASPECTOS IMPORTANTES A TENER EN CUENTA:

¿Por que es adecuada esta implementación?
Al reproducirse el vídeo en fragmentos, me permite reproducir el vídeo mientras sustituyo aquellos fragmentos que aún no han sido cargados en el buffer.
Tan pronto como tenga disponible el vídeo editado y sus correspondientes fragmentos hechos, sustituir fragmentos que aún no han sido cargados en buffer en el directorio en el que están los fragmentos que se reproducen, es transparente para el cliente consumidor.

¿Por que es importante el tamaño del buffer?
Este tiempo en el que el buffer contiene vídeo, no podrá visualizarse el vídeo editado. Sólo se verá vídeo editado a partir del momento en el que los fragmentos han sido sustituidos, pero a partir de que los fragmentos se hayan sustituido, el buffer tiene cargados una serie de fragmentos que son los fragmentos antiguos.

¿Por que es importante el tiempo que tarda en editar Ffmpeg el vídeo?
Dependiendo del tiempo que tarde en editarse el vídeo, no podrá empezar a mostrarse el vídeo editado al cliente durante su reproducción.

Si el vídeo empieza a reproducirse y Ffmpeg tarda en editar el vídeo 50 segundos, el consumidor no empezará a visualizar el vídeo editado hasta el segundo 80, ya que tendrá cargados los 30 segundos siguientes en el buffer.

¿Por que es necesario que los fragmentos sean “iguales”?
El tema de los metadatos, poco mencionado para esta prueba, pero muy importante.
Los metadatos son descritos en el fichero MPD, y este fichero no se sustituye, ya que una vez que ha comenzado a reproducirse el vídeo, va leyendo de este documento los fragmentos a pedir, si se cambiara el fragmento por uno completamente diferente, los metadatos del mismo no coinciden, y la reproducción del vídeo no sería correcta, sufriendo cortes y viéndose el vídeo mal.

¿Como ejecutar en GO herramientas que son por consola de comandos?
Al utilizar las herramientas Ffmpeg y MP4Box que son de línea de comandos con GO, se ha tenido que crear varios métodos para poder ejecutarlos desde el programa.

¿Por que es necesario utilizar la librería dash.all.min.js?
El protocolo de streaming DASH utiliza la estrategia PULL, por ello necesita que sea el reproductor quien haga la petición a el servidor, de manera que descargue el fichero manifest (MPD) y vaya pidiendo los fragmentos que completan el vídeo.

4. Funcionamiento

A continuación, se explica el funcionamiento, de la opción elegida, según lo más conveniente para llevar a cabo los objetivos indicados en el inicio de la memoria.

Para comenzar a usar la aplicación, se accede a la dirección http://localhost:8080 ya que la aplicación se está ejecutando en local, en un caso real se subiría al dominio de la empresa y se accedería mediante la web de la misma, o se acoplaría la funcionalidad en la aplicación en uso de la empresa.

Una vez accedidos en la página, se rellenará el formulario que aparece en la página principal (mostrado en la figura …)

Cuando se envía el formulario, se redirecciona a otra página, que automáticamente inicia el reproductor de vídeo, con el vídeo dividido en fragmentos (transparente para el usuario) y con la personalización aparentemente realizada.

Nuestra aplicación se divide en varios apartados:

- Página principal, que hace de interfaz para el cliente consumidor del vídeo
  Consta de una interfaz web escrita en html, que contiene un formulario, en el cual el cliente consumidor introduce los parámetros para la posterior personalización del vídeo.

  Los parámetros que el cliente introduce son:

  - Nombre
  - Apellidos
  - Segundo de inicio: segundo desde el cual se empieza a añadir los parámetros anteriores.
  - Segundo final: segundo hasta el que se añaden los parámetros

- Script que divide el vídeo en fragmentos
  Consta de un script, que tiene una serie de métodos que permiten fragmentar el vídeo a reproducir, y genera un archivo Manifest (MPD) con la descripción de los datos a enviar.
- Script que edita el vídeo
  Consta de un script, que tiene una serie de métodos que permiten añadir texto en el vídeo, para posteriormente fragmentarlo y sustituir los fragmentos en el vídeo que se encuentra reproduciéndose en el momento de la edición.

➢ Capítulo 4. Pruebas

Antes de llegar a la implementación que se ha llevado a cabo, se han probado diferentes formas de personalizar el vídeo, como las siguientes:

- **Pruebas con el protocolo DASH-MPEG:**

  - **Utilizar PERIODS en DASH.**
    Se ha intentado hacer una implementación a base de Periodos sin fragmentar, es decir, dividir el video en subvideos y reproducirlos como periodos, el problema de esta implementación es que entre periodo y periodo se hace un refresco muy rápido de la página, pero visible para el usuario.
    Esta implementación del protocolo DASH se suele usar más para integrar anuncios dentro de videos de mayor duración, como pueden ser series.

  - **Refrescar mpd cada cierto tiempo.**
    Se ha intentado implementar, refrescando cada cierto tiempo el fichero manifest (MPD) para poder ir haciendo cambios en sus fragmentos y no tener en cuenta los metadatos, pero no se ha llegado a una implementación válida ya que al refrescar el fichero mpd y cambiar los fragmentos, el cliente reproductor en HTML5 no leía correctamente el fichero mpd.

- **Pruebas con la herramienta de edición (Ffmpeg):** se ha necesitado hacer varias pruebas con esta herramienta para ver de que forma se podía editar el video en el menor tiempo posible.

  - Dividir el video en videos y editar solo el fragmento deseado y volver a unirlos: el tiempo que tarda en separar esos videos y volver a unirlos, es mayor a la opción elegida para la implementación final, que consiste en editar el video completo.

  - Se ha probado a meter el texto en durante toda la duración del video, pero se ha llegado a la conclusión que tarda menos en añadir el texto solo en los segundos que se indican en el formulario.

- **Pruebas con PlayList**: se ha implementado una play list con el video fragmentado en videos, de modo que se empezaba reproduciendo un video sin editar, mientras el segundo video se edita, para empezar a reproducirse en la play list como el segundo video, pero entre ambos videos había un pequeño corte visible para el usuario y además los tiempos de la barra de tiempo del reproductor cambiaban para cada video, siendo individuales.
  La forma de implementar la play list es con Java Script.

- **Pruebas de tiempo:**

  - **Diferencias en cuanto a donde (en que servidor) se aloja el programa:** se ha probado a ejecutar el programa en dos equipos diferentes:

    - **1er equipo:** con un hardware más potente y con el sistema operativo Ubuntu 16.04 para en añadir los parámetros de texto en el vídeo durante 10 segundos, tarda unos 45 segundos.
    - **2º equipo:** con un hardware algo menos potente que el anterior y con el mismo sistema operativo, ejecutando exactamente el mismo comando, se tarda 1,15 segundos, es decir, medio minuto más.

    Conclusiones: es de gran importancia decidir en que servidor se va a ejecutar la aplicación ya que depende mucho los tiempos en los que permite empezar a reproducir el video editado.

  - **Importancia del Buffer:** el tamaño del buffer tiene un papel importante en nuestra implementación, ya que una vez haya terminado de editar el video, no se podrá empezar a mostrar el video editado hasta después del tiempo que tenga almacenado el buffer, en mi caso 30 segundos, se ha intentado buscar la forma de disminuir ese tiempo ya que cuanto menos sea el tamaño del buffer, antes se podrá empezar a ver el video editado, pero no se ha podido encontrar en el tiempo estipulado para el desarrollo de este proyecto.

- **Pruebas con los metadatos:**

  - Comprender el uso de los metadatos y como afectan a la hora de fragmentar el vídeo con diferentes metadatos y en su posterior reproducción: se ha necesitado un periodo de aprendizaje acerca de cómo afectan los metadatos a la hora de editar un video y reproducirlo con el protocolo DASH y fragmentos.
    Se ha centrado sobre todo en dos de sus metadatos, ya que la importancia de la calidad del video, y los fotogramas por segundo del mismo, afectan a la hora de reproducir el video, si estos metadatos cambian, el video se ejecuta de manera irregular y con problemas de pixelado.

  - Se ha utilizado la herramienta de consola de Linux “mediainfo” para ver los metadatos de los videos y la herramienta Ffmpeg para editarlos.

- **Pruebas para reproducir el vídeo por partes, mediante HTML:** se ha intentado reproducir el video por partes a nivel de html escribiendo un atributo en la etiqueta video con los segundos que se quería reproducir en cada parte, pero con esta opción no ser modifica el video ni la manera de enviarlo, solo su reproducción, opción que no nos sirve.

➢ Capítulo 5. Seguimiento y control

Tarea	Planificado	Dedicado	Desviación
Gestión del TFG	80 horas	95 horas	+19%
1.1. Generación DOP (Documento Objetivos Proyecto)	5 horas	8 horas	+60%
1.2 Planificación temporal inicial	2,5 horas	5 horas	+100%
1.3 Planificación temporal	3,5 horas	10 horas	+85%
1.4 EDT	3 horas	5 horas	+66%
1.5 Gantt	5 horas	5 horas	0%
2.1 Gestión de tiempo	3 horas	2 horas	+66%
2.2 Reuniones	5 horas	5 horas	0%
3.1. Estudio previo	3 horas	5 horas	+66%
3.2. Estructura	3 horas	5 horas	+66%
3.3. Redacción memoria	30 horas	40 horas	+33%
3.4. Revisión	6 horas	6 horas	0%
4.1. Crear presentación	5 horas	?	
4.2. Preparar defensa	5 horas	?	
4.3. Defensa ante tribunal	1 hora	?	
Análisis	75 horas	87 horas	+16%
1. Estado del Arte	20 horas	25 horas	+25%
2. Formatos de vídeo	5 horas	5 horas	0%
3. Librerías / Herramientas de edición	10 horas	10 horas	0%
4. Formas de enviar vídeo	37 horas	45 horas	+22%
5. Requisitos del cliente	3 horas	2 horas	-33%

Tabla5. Gestión de tiempo

Formación	80 horas	90 horas	+12%
1. Lenguaje GO	20 horas	25 horas	+25%
2. Librerías / Herramientas elegidas	15 horas	15 horas	0%
3. Forma de enviar el vídeo elegida	45 horas	50 horas	+25%
Implementación	40 horas	25 horas	-37%
1. Configuración del entorno de desarrollo	5 horas	5 horas	0%
2. Parte I: edición de vídeo	10 horas	5 horas	-50%
3. Parte II: envío de vídeo	20 horas	10 horas	-50%
4. Parte III: Interfaz cliente (reproductor)	5 horas	5 horas	0%
Pruebas	20 horas	30 horas	+50%
1. Diseño de pruebas	6 horas	10 horas	+66%
2. Ejecución de pruebas	9 horas	15 horas	+66%
3. Corrección de errores	5 horas	5 horas	0%
Conclusiones	2,5 horas	2 horas	-20%
Bibliografía	5 horas	4 horas	-20%

Tabla5.(Continuación) Gestión de tiempo

➢ Capítulo 6. Conclusiones y futuras mejoras

Tras la realización de este proyecto se ha conseguido una implementación con unas características determinadas, que son lo que se pedía en los objetivos del proyecto, pero para un uso real, se necesita otro segundo periodo de investigación para mejorar la eficiencia en cuanto a tiempo de edición, ya que, si se tratara de videos publicitarios, serian video de menos de 3 minutos, que es para el tiempo que se ha estimado que la aplicación funciona de manera óptima.

En ese segundo periodo se debería estudiar también la manera de disminuir el tiempo de carga en buffer ya que es una pauta importante a la hora de poder empezar a reproducir el video editado cuanto antes.

Se ha notado que la planificación no ha sido del todo correcta a la hora de llevarlo a cabo, ya que se ha perdido un gran tiempo en modificar cosas ya hechas o estudiadas.

Al ser un proyecto de “investigación” ha resultado más difícil realizar la planificación porque no se sabía con exactitud lo que posteriormente se iba a implementar y de que manera, por lo que tampoco se ha podido hacer una buena planificación en cuanto a la parte de implementación del mismo.

Se ha necesitado también invertir bastante tiempo en entender cómo funciona la empresa Viwom, ya que en las practicas se había estado trabajando con diferentes materiales y con un enfoque diferente a lo que después se ha realizado en el proyecto.

Bajo mi punto de vista, este proyecto hubiera sido más fácil de realizar si se hubiera contado con la ayuda de personas externas, ya que ha habido veces que me he atascado en cosas por falta de conocimiento en otros aspectos ajenos al proyecto.

También ha habido que invertir tiempo en aprender el lenguaje de programación con el que se ha llevado a cabo la prueba de concepto, lo que hubiera sido más fácil hacerlo con un lenguaje ya conocido, pero por petición de la empresa se ha tenido que hacer con GO.

➢ Capítulo 7. Bibliografía

[1] http://demo.rtcvid.net/tts/

[2] http://videopersonalization.net/

[3] https://www.vidyard.com/platform/personalizedvideo/custom_id=5dQXBdwzA6euSdAgJ3VgBP

[4] https://ffmpeg.org/about.html

[5] https://www.internetya.co/hosting-para-moodle-en-colombia-por-que-usar-una-cdn/

[6] http://wwwhtml5rockscom.readthedocs.io/en/latest/content/tutorials/streaming/multimedia/en/

[7] http://stackoverflow.com/questions/33389001/what-is-blob-in-the-video-src-bloburl

[8] https://www.w3schools.com/html/html5_video.asp

[9] https://developer.mozilla.org/es/docs/Web/HTML/Formatos_admitidos_de_audio_y_video_en_html5

[10] https://github.com/zencoder/go-dash

[11] https://golanglibs.com/top?q=mpeg l

[12] https://godoc.org/github.com/zencoder/go-dash/mpd

[13] https://bitmovin.com/dynamic-adaptive-streaming-http-mpeg-dash/

[14] http://www.hbbtv-developer.com/site/blog/?p=879

[15] http://upcommons.upc.edu/bitstream/handle/2099.1/15461/memoria.pdf?sequence=4&isAllowed=y

[16] https://www.scribd.com/doc/46514468/Manual-basico-del-comando-ffmpeg

[17] https://godoc.org/github.com/xdave/ff

[18] https://docs.google.com/document/d/1_Y9xCEMj5S-7rv2ooHpZNH15JgRT5iM742gJkw5LtmQ/edit

[19] https://github.com/GoClipse/goclipse/blob/latest/documentation/UserGuide.md#configuration

[20] http://www.atc.uniovi.es/teleco/5tm/archives/8streaming.pdf

[21] http://stackoverflow.com/questions/38375145/golang-executing-a-command-with-its-arguments

[22] http://www.vocalcom.com/products/personalized-videos-platform/

[23] https://www.sitepoint.com/html5-video-fragments-captions-dynamic-thumbnails/

[24] https://gist.github.com/CMCDragonkai/6bfade6431e9ffb7fe88

[25] http://thekuroko.com/2011/06/20/http-dynamic-streaming-getting-started/

[26] http://stackoverflow.com/questions/10110479/upload-a-video-file-by-chunks

[27] http://stackoverflow.com/questions/24976123/streaming-a-video-file-to-an-html5-video-player-with-node-js-so-that-the-video-c

[28] http://html5-demos.appspot.com/static/media-source.html

[29] https://desarrolloweb.com/manuales/manual-streaming-video.html
[30] https://developer.apple.com/library/content//documentation/NetworkingInternet/Conceptual/StreamingMediaGuide/UsingHTTPLiveStreaming/UsingHTTPLiveStreaming.html
[31] http://stackoverflow.com/questions/28565175/send-video-over-http-http-multipart-request
[32] http://www.dreamincode.net/forums/topic/347938-a-new-webcam-api-tutorial-in-c-for-windows/
[33] https://www.html5rocks.com/en/tutorials/video/basics/
[34] https://www.sitepoint.com/html5-video-fragments-captions-dynamic-thumbnails/
[35] https://alastaira.wordpress.com/2014/10/18/grabbing-and-converting-adobe-streaming-video-fragments/ fragmentos con adobe con f4f
[36] http://www.nurkiewicz.com/2015/06/writing-download-server-part-iii.html
[37] https://www.w3.org/2008/WebVideo/Fragments/wiki/HTTP_Examples
[38] https://en.wikipedia.org/wiki/List_of_HTTP_header_fields
[39] http://stackoverflow.com/questions/24976123/streaming-a-video-file-to-an-html5-video-player-with-node-js-so-that-the-video-c
[40] http://stackoverflow.com/questions/13043816/html5-video-and-partial-range-http-requests
[41] http://stackoverflow.cm/queostions/35411585/go-lang-first-frame-of-video con go
[42] http://cloudinary.com/blog/plug_and_play_adaptive_bitrate_streaming_with_hls_and_mpeg_dash
[43] https://modernweb.com/building-an-online-video-player-with-dash-264/
[44] https://blog.streamroot.io/encode-multi-bitrate-videos-mpeg-dash-mse-based-media-players-22/
[45] http://subtitling.irt.de/
[46] https://stackoverflow.com/questions/41460865/mpeg-dash-with-live-stream
[47] https://dashif.org/reference/players/javascript/v2.4.0/samples/dash-if-reference-player/
[48] http://cdn.dashjs.org/latest/jsdoc/moduleMediaPlayer.html#setBufferTimeAtTopQualityanchor
[49] https://github.com/Dash-Industry-Forum/dash.js/issues/1898

➢ Anexos

➢ Anexo1: Análisis y estudio de los formatos de vídeo

Principales formatos:

WebM:
Formato contenedor: WebM, (versión restringida del contenedor Matroska)
Códecs: vídeo VP8 y audio Vorbis.
Compatibilidad: Firefox, Chrome, Opera, posibilidad de agregar mediante un add-on Internet Explorer y Safari.
Licencia: gratuita.
MIME: vídeo/webm y audio/webm

Ogg Theora Vorbis
Formato contenedor: Ogg
Códecs: vídeo Theora y audio Vorbis
Compatibilidad: Firefox, Chrome y opera, Safari mediante add-on. No Internet explorer.
Licencia: gratuita
MIME: audio/ogg, vídeo/ogg y application/ogg

MP4 H.264
Formato contenedor: MP4
Códecs: vídeo H.264 y audio ACC o MP3
Compatibilidad: Internet Explorer, safari y Chrome, No Chronium y Opera, Firefox en proceso.
Licencia: NO libre.
MIME: vídeo/mp4

Otros:

WAVE PCM
Formato contenedor: WAVE
Códecs: de audio PCM
tienen la extensión .wav
MIME: audio/wave, audio/wav, audio/x-wav, audio/x-pn-wav

HTML Vídeo- soporte de navegadores:

Navegador	MP4	WebM	Ogg
Internet Explorer	SI	NO	NO
Chrome	SI	SI	SI
Firefox	SI	SI	SI
Safari	SI	NO	NO
Opera	SI (para Opera 25)	SI	SI

➢ Anexo2: Instalación y configuración del entorno de desarrollo

INSTALACIÓN DEL ENTORNO DE DESARROLLO Y LAS HERRAMIENTAS NECESARIAS.

Se ha trabajado con el entorno de desarrollo de eclipse, sobre la plataforma de Linux, distribución Ubuntu Desktop 16.04.

Para instalar goclipse se necesitan los siguientes requerimientos:

- Java VM versión 8
- Eclipse 4.6 (Neon)
- CDT 9.0

Se instala Java version8:

1. Escribimos los comandos:

```
sudo add-apt-repository ppa:webupd8team/java
sudo apt-get update
sudo apt-get install oracle-java8-installer
```

Se instala **Eclipse** descargándolo de la página oficial del mismo: https://www.eclipse.org/downloads/download.php?file=/oomph/epp/neon/R2a/eclipse-inst-linux64.tar.gz

Elegiré el eclipse IDE for C/C++

INSTALACIÓN DE GO Y ACOPLAMIENTO CON EL IDE (Eclipse).

Una vez descargado e instalado en el entorno de desarrollo, necesitaremos tener instaladas las herramientas de Go, que hay un guion de cómo hacerlo en el siguiente enlace:

**** Instalación manual ****

Documentación acerca de cómo instalarlo: https://golang.org/doc/install

1. Descargamos el archivo de go del enlace: https://golang.org/dl/
2. Lo descomprimimos en /usr/local/go con el comando:
   `tar -C /usr/local -xzf go$VERSION.$OS-$ARCH.tar.gz`
3. Ahora habrá que añadir en la variable de entorno PATH el directorio /usr/local/go/bin con el siguiente comando:
   `export PATH=$PATH:/usr/local/go/bin`
4. Añadir para que se haga persistente la variable de entorno editando /etc/enviroment
   1. PATH=……..:/usr/local/go/bin
   2. salir y escribir source /etc/enviroment

****Instalación con apt-get****

Bastará con escribir el siguiente comando en la terminal:

sudo apt-get install golang-go

Además, el IDE requiere de funcionalidad, para ello, le instalaremos los siguientes **plugins:**

→ **goclipse:** para funcionar con GO desde el entorno de desarrollo.

Se instala en el entorno de desarrollo:

- Pulsando en la pestaña Help → Install New Software …
- Hacemos click en Add, y en el apartado de URL escribimos: http://goclipse.github.io/releases/ y pulsamos OK.

  Hay que añadir la variable GOPATH, que será donde se encuentre el proyecto.: **GOPATH="/home/viwom.inet/airuzubieta/goProject"**, es decir, un directorio que no es el mismo que el de instalación.

Ahora vamos a instalar algunas de las herramientas de ayuda que tiene go para programar.

Situados en la carpeta del proyecto al cual le queremos añadir estas ayudas, es decir en el mismo directorio que hemos puesto el path, y ahí en el directorio “src”

Herrramientas de go: sudo apt-get install golang-golang-x-tools

Guru:
$ go get golang.org/x/tools/cmd/guru
$ go build golang.org/x/tools/cmd/guru

godef:
go get github.com/rogpeppe/godef
go build github.com/rogpeppe/godef

gofmt: si instalamos go haciendo sudo apt-get install golang-go nos viene instalada gofmt y godoc.

Libreria ffmpeg: go get github.com/xdave/ff

Gocode: go get github.com/nsf/gocode
go build github.com/nsf/gocode

En el editor de eclipse, añadiremos donde se encuentran las librerías añadidas, ej:
Windows→ preferences → Go → Tools y en gocode examinar y seleccionar el que está en /usr/local/go/bin/gocode

➢ Anexo3: Puesta en producción

Una vez finalizada la aplicación, se aloja en un servidor Ubuntu, y se ejecuta.
El servidor web programado en GO está escuchando peticiones por el puerto 8080.

En el sistema se deben tener instalado lo siguiente:

- Go: para instalarlo se escribe en la terminal:

  sudo apt-get install golang

- Ffmpeg:

  sudo apt-get install ffmpeg

- MP4Box:

  sudo apt-get install gpac

Una vez instaladas las herramientas se hace alguna prueba para comprobar que funcionan correctamente y no falta nada por configurar.

➢ Anexo4: Reuniones

Reuniones con Tutor Universidad: Eloy Mata

1. **Reunión 1:** Pequeña introducción y guiá para empezar
   En esta reunión se…

2. **Reunión 2:** Se quiere empezar a escribir documentación y aclarar aspectos.
   En esta reunión ...

3. **Reunión 3:** Se quiere concretar los primeros entregables.
   En esta reunión Eloy pide los siguientes entregables escritos para una primera corrección.

   - Utilizar la metodología de trabajo Iterativa incremental.
   - Documento Objetivos del Proyecto(DOP)
   - Descomposición de Tareas con estimación del tiempo (diagrama EDT)
   - Enumeración y descripción de las mismas.
   - Enumerar y explicar las formas encontradas de hacer la personalización.
   - Enumerar y explicar posibles formatos de video para la web (compatibilidades...)
   - Hacer pruebas de incrustar texto en el video. (calcular tiempos con diferentes videos).

   Se pide que la entrega sea el Lunes, 20 Marzo 10:00.

4. **Reunión 4:** Corrección de la primera entrega de documentación.
   En esta reunión hemos tratado de corregir la primera entrega realizada, y mejorar la estructura de la misma.

5. **Reunión 5:** Corrección de la segunda entrega de la documentación.
   En esta reunión se ha tratado de corregir aspectos de la escritura y la expresión de la redacción, también de algunos de los contenidos.

6. **Reunión 6:** Corrección de la tercera entrega de la documentación y muestra de la aplicación.

7. **Reunión 7:** Ultima corrección y firma para poder entregar el TFG.

Reuniones con Tutor Empresa: Jorge Garcés

1. **Reunión 1:** Descripción del proyecto a realizar

2. **Reunión 2:** Concreción de diferentes pautas y dudas

3. **Reunión 3:** Muestra de algunas de las opciones encontradas

4. **Reunión 4:** revisión de tareas y sugerencias de mejora.

5. **Reunión 5:** presentación del trabajo finalizado.